TUGAS 6 PRAKTIKUM PBO

KELAS ABSTRAKSI, INTERFACE, & METACLASS

OLEH:

MUHAMMAD RAIHAN PUTERANDA

121140089

RB

PROGRAM STUDI TEKNIK INFORMATIKA

JURUSAN TEKNOLOGI PRODUKSI DAN INDUSTRI

INSTITUT TEKNOLOGI SUMATERA

LAMPUNG SELATAN

2023

RINGKASAN MATERI

A. Kelas Abstraksi

Kelas abstraksi atau kelas abstrak berarti kelas yang masih belum jelas implementasi-nya atau abstrak sehingga tidak bisa di-instansiasikan atau dibuat objeknya. Untuk bisa disebut kelas abstrak, setidaknya perlu 1 metode abstrak. Pembuatan kelas abstrak sendiri harus mengambil kelas ABC terlebih dahulu pada modul abc, begitu pula untuk membuat suatu metode abstrak perlu mengambil decorator @abstractmethod dari modul abc.

Metode atau fungsi abstrak dari sebuah kelas abstrak tidak dapat diimplementasikan pada kelasnya sendiri melainkan pada kelas turunannya, sehingga jika sebuah metode abstrak dibuat dan kelas turunannya memiliki fungsi tersebut, maka harus diimplementasikan di kelas tersebut.

Kelas abstraksi juga boleh memiliki sebuah konstruktor dan fungsi konkrit asalkan ada satu buah fungsi abstrak didalamnya.

Contoh dari kelas abstrak:

```python
from abc import ABC, abstractmethod

class Game(ABC):
    def __init__(self, nama, genre):
        self.nama = nama
        self.genre = genre

    @abstractmethod
    def update_game(self):
        pass

    @abstractmethod
    def render_game(self):
        pass

    def mulai_game(self):
        print("Game dijalankan!")

class SanAndreas(Game):
    def __init__(self, nama, genre, ukuran):
        super().__init__(nama, genre)
        self.ukuran = ukuran

    def update_game(self):
        size_update = 50
        print(f"Game akan diupdate dengan ukuran {size_update} GB")

    def render_game(self):
        if (self.ukuran > 10):
            print("Waktu loading sekitar 5 menit")

# main

GTA = Game("San Andreas", "Adventure")
```

```
Traceback (most recent call last):
  File "d:\Tugas\Kuliah\SMT 4\OOP\TEST PRAK 6.py", line 34, in <module>
    GTA = Game("San Andreas", "Adventure")
TypeError: Can't instantiate abstract class Game with abstract methods render_game, update_game
```

Dari contoh tersebut, GTA tidak bisa dibuat menjadi objek “Game” karena merupakan suatu kelas abstraksi, dan juga fungsi update dan render game diturunkan kepada “SanAndreas” karena implementasi dari berbagai macam game tentu berbeda

B. Interface

Interface berarti suatu fitur yang dimiliki sebuah kelas, dimana didalamnya terdapat kumpulan fungsi yang dapat digunakan oleh kelas yang memiliki interface tersebut. Interface hanya menyediakan daftar kegunaan yang dapat dilakukan suatu kelas tanpa implementasi sehingga semua fungsi didalamnya bersifat abstrak. Sama halnya seperti kelas abstrak, implementasinya dilakukan oleh kelas yang memiliki interface tersebut (kelas turunan). Interface dapat diterapkan di python secara informal dengan fungsi yang dapat di override dan tidak dipaksakan atau tidak harus dimiliki kelas yang memiliki interface tersebut. Dapat juga diterapkan secara formal yakni menggunakan kelas abstrak beserta fungsi abstraknya yang dipaksakan kepada kelas yang memiliki interface tersebut.

Contoh dari interface:

```python
from abc import ABC, abstractmethod

class Aksi(ABC):
    @abstractmethod
    def gerak(self):
        pass

    @abstractmethod
    def serang(self):
        pass

    @abstractmethod
    def heal(self):
        pass
```

Dari contoh diatas, misalkan dalam sebuah game pasti memiliki sebuah karakter yang dapat di kontrol, sehingga diberikan sebuah interface berupa aksi yang memiliki berbagai tindakan yang bisa dilakukan berbagai macam karakter. Berbagai karakter dapat melakukan tindakan yang berbeda mulai dari gerakan, serangan, sampai cara menyembuhkan diri, sehingga penggunaan interface diperlukan untuk hal semacam ini.

C. Metaclass

Metaclass berarti suatu kelas khusus yang hierarki nya lebih tinggi daripada suatu kelas sehingga menjadikan suatu kelas sebagai instansi dari metaclass dan sering disebut sebagai kelas dari kelas. Metaclass mengontrol perilaku yang terdapat pada suatu kelas seperti penambahan atribut atau metode. Penerapannya yang paling umum yakni menggunakan "type" yang merupakan bawaan python atau dengan menggunakan parameter metaclass.

Contoh dari metaclass:

```
def mulai_film():
    print("Film diputar")

babylon = type("Film", (), {"genre": "Drama", "play": mulai_film})

print(f"Kelas dibuat bernama {babylon}")
print(f"Genre film ini adalah {babylon.genre}")
babylon.play()
```

```
Kelas dibuat bernama <class '__main__.Film'>
Genre film ini adalah Drama
Film diputar
```

Dari contoh tersebut, penggunaan metaclass akan membuat suatu kelas "Film" beserta atribut "Genre" dan metode "Play" yang akan mengambil fungsi dari mulai_film.

Contoh dengan parameter metaclass:

```
class Film(type):
    def __new__(cls, name, bases, attrs):
        attrs["genre"] = "Drama"
        return super().__new__(cls, name, bases, attrs)

class Babylon(metaclass = Film):
    pass

print(Babylon.genre) # Drama
```

Dari contoh diatas, kelas "Babylon" menetapkan "Film" sebagai metaclassnya sehingga "Film" bisa mengontrol perilaku didalam "Babylon", dalam hal ini menambahkan atribut "Genre" tanpa harus diimplementasikan di kelasnya. Parameter cls sendiri untuk menetapkan kelas yang akan dikontrol, name untuk memberi nama kelas, bases untuk mengetahui superclass dari kelas yang dikontrol, dan attrs untuk membuat atribut dan metode.

KESIMPULAN

- **Apa itu interface dan kapan kita perlu memakainya?**
- **Apa itu kelas abstrak dan kapan kita perlu memakainya? Apa perbedaannya dengan interface?**
- **Apa itu kelas konkrit dan kapan kita perlu memakainya?**
- **Apa itu metaclass dan kapan kita perlu memakainya? Apa bedanya dengan inheritance biasa?**

Jawaban:

- Interface adalah suatu fitur berisi kumpulan konsep dari fungsi yang akan diimplementasikan pada suatu program (kelas). Interface dipakai saat pengembang ingin menetapkan suatu fitur yang memiliki fungsi tertentu pada program yang berbeda-beda. Misalnya saat mengembangkan sekuel dari sebuah game, mungkin ada beberapa fitur dari game sebelumnya yang ingin diterapkan dan ingin diubah cara kerjanya, maka menggunakan sebuah interface.

- Kelas abstrak adalah kelas yang belum ada implementasinya dan tidak bisa dibentuk suatu objek, digunakan saat pengembang ingin menerapkan aturan dasar yang harus ditetapi suatu program. Perbedaannya dengan interface sendiri yakni kelas abstrak memungkinkan untuk membuat fungsi konkrit atau fungsi umum yang diterapkan pada kelasnya sendiri, sedangkan interface fungsinya harus abstrak semua tidak boleh ada fungsi konkrit.

- Kelas konkrit adalah kelas yang dapat dinstansiasi dan dibuat objeknya, digunakan saat pengembang ingin membuat suatu tugas atau komponen tertentu yang spesifik atau saat memiliki sebuah kelas abstrak untuk dibuat implementasinya.

- Metaclass adalah kelas dari kelas yang berarti pengontrol perilaku dari suatu kelas seperti atribut dan metodenya, penggunaan metaclass justru dapat membuat kode menjadi lebih rumit tapi bisa digunakan saat pengembang ingin menata kode agar lebih baik dan elegan serta untuk menerapkan desain atau pola khusus pada suatu kelas. Bedanya dengan inheritance biasa yakni metaclass berfokus pada pengontrolan sebuah kelas itu sendiri mulai dari perilaku sampai ke pembuatannya, sedangkan inheritance lebih fokus ke pewarisan atribut dan metode saja.

DAFTAR PUSTAKA

[1] A. Krishna, "Abstract Classes in Python," 19 Maret 2021. [Online]. Available: https://www.geeksforgeeks.org/abstract-classes-in-python/. [Accessed 4 Maret 2023].

[2] W. Murphy, "Implementing an Interface in Python," [Online]. Available: https://realpython.com/python-interface/. [Accessed 4 Maret 2023].

[3] S. George, "Python MetaClasses," 18 Agustus 2020. [Online]. Available: https://www.geeksforgeeks.org/python-metaclasses/. [Accessed 4 Maret 2023].

[4] J. Sturtz, "Python Metaclasses," [Online]. Available: https://realpython.com/python-metaclasses/. [Accessed 4 Maret 2023].

[5] M. Das, "Advanced Python Topics: Metaclasses vs Inheritance," 24 Februari 2023. [Online]. Available: https://python.plainenglish.io/advanced-python-topics-metaclasses-vs-inheritance-a39154ebb6f2. [Accessed 4 Maret 2023].