LAPORAN PRAKTIKUM
PEMROGRAMAN BERORIENTASI OBJEK
MODUL 6

Oleh :

David Gunawan (121140062)

Program Studi Teknik Informatika

Institut Teknologi Sumatera

2023

Pembahasan

dalam pemrograman berorientasi objek (PBO) terdapat 3 konsep penting yaitu kelas abstrak, interface, dan metaclass. Ketiga konsep ini sangat penting dalam PBO, terutama ketika membuat kode yang dapat digunakan kembali dan mudah dimodifikasi. Kelas abstrak, interface, dan metaclass adalah konsep yang cukup kompleks, oleh karena itu kami mencari beberapa referensi untuk membantu kami memahami konsep tersebut dengan baik. Beberapa referensi yang kami gunakan antara lain: "Head First Design Patterns" oleh Eric Freeman dan Elisabeth Freeman, "Python for Data Science Handbook" oleh Jake VanderPlas, dan "Fluent Python" oleh Luciano Ramalho.

KELAS ABSTRAK

Kelas abstrak adalah kelas yang tidak dapat diinstansiasi langsung. Kelas ini hanya digunakan sebagai kerangka atau cetakan untuk membuat kelas turunan. Kelas abstrak digunakan untuk mewakili konsep umum dari kelas-kelas turunan yang lebih spesifik. Kelas abstrak sendiri tidak dapat diinstansiasi karena masih memiliki metode yang tidak diimplementasikan.

Untuk membuat kelas abstrak, kita perlu mengimport modul abc (Abstract Base Class). Modul ini menyediakan kelas ABC (Abstract Base Class) dan decorator @abstractmethod yang digunakan untuk mendefinisikan metode abstrak. Sebuah kelas abstrak harus memiliki minimal satu metode abstrak, dan setiap metode abstrak harus diimplementasikan di kelas turunannya.

Berikut adalah contoh Implementasi kelas abstrak pada bahasa pemrograman python :

```python
from abc import ABC, abstractmethod

class bentuk(ABC):
    @abstractmethod
    def luas(self):
        pass

class persegiempat(bentuk):
    def __init__(self, panjang, lebar):
        self.panjang = panjang
        self.lebar = lebar
    def luas(self):
        return self.panjang * self.lebar

class lingkaran(bentuk):
    def __init__(self, jarijari):
        self.jarijari = jarijari
    def luas(self):
        return 3.14 * self.jarijari ** 2
```

Pada contoh di atas, kita membuat kelas abstrak bernama bentuk yang memiliki satu metode abstrak bernama luas(). Kemudian, kita membuat dua kelas turunan dari bentuk yaitu persegiempat dan lingkaran yang mengimplementasikan metode luas() sesuai dengan kebutuhan masing-masing.

INTERFACE

Interface adalah sebuah blueprint atau cetakan yang digunakan untuk menentukan perilaku kelas. Interface mendefinisikan suatu kontrak atau kesepakatan antara kelas yang menggunakan interface dengan kelas yang mengimplementasikan interface tersebut. Interface juga dapat disebut sebagai sekumpulan metode abstrak. Interface tidak memiliki implementasi atau kode di dalamnya, melainkan hanya berisi nama metode dan tipe data parameter yang diterima serta tipe data return-nya.

Interface digunakan untuk memastikan bahwa kelas-kelas turunan yang mengimplementasikan interface memiliki perilaku yang sama. Interface sangat berguna ketika kita ingin membuat beberapa kelas yang berbeda namun memiliki perilaku yang sama. Dengan menggunakan interface, kita dapat memastikan bahwa setiap kelas turunan dari interface tersebut memiliki metode yang sama dan sesuai dengan kontrak yang telah disepakati.

Untuk membuat interface pada bahasa pemrograman Python, kita dapat menggunakan modul abc (Abstract Base Class) seperti pada pembuatan kelas abstrak. Namun, untuk membuat sebuah interface, kita hanya perlu mendefinisikan metode abstrak tanpa adanya implementasi.

Berikut adalah contoh implementasi interface pada bahasa pemrograman Python:

```python
from abc import ABC, abstractmethod

class hewan(ABC):
    @abstractmethod
    def suara(self):
        pass

class Anjing(hewan):
    def suara(self):
        return "guk guk!"

class Kucing(hewan):
    def suara(self):
        return "Meow!"
```

Pada contoh di atas, kita membuat sebuah interface bernama Animal yang memiliki satu metode abstrak bernama speak(). Kemudian, kita membuat dua kelas turunan dari Animal yaitu Dog dan Cat yang mengimplementasikan metode speak() sesuai dengan perilaku masing-masing.

KELAS KONKRET

Kelas konkret adalah kelas yang dapat diinstansiasi langsung. Kelas ini merupakan turunan dari kelas abstrak atau mengimplementasikan interface. Kelas konkret memiliki implementasi dari semua metode yang ada di kelas abstrak atau interface yang diimplementasikannya.

Kelas konkret digunakan untuk membuat objek yang dapat digunakan langsung. Kelas konkret sering kali digunakan untuk menghasilkan objek dari kelas-kelas abstrak atau interface.

Berikut adalah contoh implementasi kelas konkret pada bahasa pemrograman Python:

```
class persegi(bentuk):
    def __init__(self, sisi):
        self.sisi = sisi
    def luas(self):
        return self.sisi ** 2
```

Pada contoh di atas, kita membuat sebuah kelas konkret bernama persegi yang merupakan turunan dari kelas abstrak bentuk dan mengimplementasikan metode luas() sesuai dengan kebutuhan kelas tersebut.

METACLASS

Metaclass adalah kelas yang digunakan untuk membuat kelas. Metaclass berperan sebagai pengontrol atau pemilik kelas. Ketika sebuah kelas dibuat, Python akan mencari metaclass yang terkait dengan kelas tersebut. Jika tidak ada metaclass yang ditentukan, Python akan mencari metaclass dari kelas induk atau menggunakan tipe bawaan.

Metaclass digunakan untuk mengontrol pembuatan kelas dan memungkinkan pengguna untuk menentukan perilaku kelas yang berbeda dari perilaku kelas standar. Metaclass sering digunakan untuk mengimplementasikan fitur-fitur seperti validasi input, caching, dan lain-lain.

Namun, penggunaan metaclass tidak umum dalam pemrograman berorientasi objek dan hanya digunakan dalam kasus-kasus khusus.

Berikut adalah contoh sederhana penggunaan metaclass pada Python:

```
class MyMeta(type):
    def __new__(cls, name, bases, attrs):
        new_attrs = {}
        for attr_name, attr_value in attrs.items():
            if callable(attr_value):
                new_attrs[attr_name.upper()] = attr_value
            else:
                new_attrs[attr_name] = attr_value
        return super().__new__(cls, name, bases, new_attrs)

class MyClass(metaclass=MyMeta):
    def hello(self):
        print("Hello World!")

obj = MyClass()
obj.HELLO()
```

Pada contoh di atas, MyMeta adalah metaclass yang digunakan untuk membuat kelas MyClass. Ketika MyClass dibuat, metaclass MyMeta dijalankan untuk mengubah atribut hello menjadi HELLO dan memasukkannya ke dalam kelas. Kemudian, MyClass diinstansiasi dan metode HELLO dipanggil untuk mencetak "Hello World!". Dengan menggunakan metaclass, kita dapat mengubah perilaku kelas secara dinamis, misalnya dengan mengubah nama metode atau menambahkan atribut baru.

Kesimpulan

1. Apa itu interface dan kapan kita perlu memakainya?

Interface adalah suatu abstraksi yang digunakan untuk mendefinisikan suatu perilaku tanpa harus menentukan implementasi detail dari perilaku tersebut. Dalam bahasa pemrograman, interface biasanya berisi sekumpulan definisi fungsi atau metode tanpa mengimplementasikan detail dari fungsi atau metode tersebut. Interface digunakan untuk mempermudah pengembangan aplikasi karena memungkinkan beberapa kelas yang berbeda untuk mengimplementasikan perilaku yang sama secara konsisten. Interface dapat digunakan ketika kita ingin memaksa kelas-kelas tertentu untuk mengimplementasikan perilaku yang sama atau ketika kita ingin menghindari ketergantungan pada implementasi kelas tertentu.

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

Kelas abstrak adalah kelas yang tidak dapat diinstansiasi dan hanya digunakan sebagai kerangka untuk kelas turunannya. Kelas abstrak digunakan untuk mendefinisikan suatu perilaku yang harus diimplementasikan oleh kelas turunannya. Perbedaan utama antara kelas abstrak dan interface adalah bahwa kelas abstrak dapat memiliki implementasi untuk beberapa atau semua metodenya, sedangkan interface tidak memiliki implementasi sama sekali.

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

Kelas konkret adalah kelas yang dapat diinstansiasi dan memiliki implementasi untuk semua metodenya. Kelas konkret digunakan ketika kita ingin membuat objek yang dapat digunakan langsung tanpa perlu melakukan implementasi tambahan.

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

Metaclass adalah kelas yang digunakan untuk membuat kelas baru. Metaclass dapat digunakan untuk memodifikasi perilaku kelas secara dinamis pada waktu pembuatan kelas. Perbedaan utama antara metaclass dan inheritance biasa adalah bahwa metaclass memungkinkan kita untuk memodifikasi perilaku kelas secara dinamis pada waktu pembuatan kelas, sedangkan inheritance biasa hanya memungkinkan kita untuk mewarisi perilaku dari kelas induk secara statis. Metaclass perlu digunakan ketika Anda ingin membuat sebuah kelas dengan perilaku khusus yang tidak bisa diatur dengan inheritance biasa atau ketika Anda ingin membuat banyak kelas dengan perilaku yang sama secara otomatis.

Daftar Pustaka

Rossum, G. V. (2003). Python tutorial. Technical report.

Dasgupta, A. (2017). Python for Everybody: Exploring Data in Python 3. CreateSpace Independent Publishing Platform.

Hunt, C., & Augenstein, R. (2014). The Pragmatic Programmer: From Journeyman to Master. Addison-Wesley Professional.

GeeksforGeeks. (2022). Python - Classes and Objects. https://www.geeksforgeeks.org/python-classes-and-objects/.

Real Python. (2022). Understanding Python Metaclasses. https://realpython.com/python-metaclasses/.