LAPORAN RINGKASAN MATERI
PEMROGRAMAN BERORIENTASI OBJEK
(RB)

Oleh :

Ferdinand Zulvan Lindan (121140170)

Program Studi Teknik Informatika

Institut Teknologi Sumatera

2023

Ringkasan

A. . Kelas Abstrak

1. Kelas Abstrak adalah kelas yang tidak dapat diinstansiasi dan berfungsi sebagai kerangka kerja untuk kelas-kelas turunannya. Kelas Abstrak dapat memiliki metode abstrak (tidak memiliki implementasi) dan metode non-abstrak. Beberapa poin penting tentang kelas abstrak adalah:

2. Tidak dapat diinstansiasi: Anda tidak dapat membuat objek langsung dari kelas abstrak. Namun, Anda dapat membuat objek dari kelas turunan yang mengimplementasikan metode-metode abstrak yang didefinisikan dalam kelas abstrak tersebut.

3. Memiliki metode abstrak: Metode abstrak hanya dideklarasikan di dalam kelas abstrak tanpa memiliki implementasi. Metode abstrak harus diimplementasikan oleh kelas turunan.

4. Dapat memiliki metode non-abstrak: Kelas abstrak juga dapat memiliki metode non-abstrak, yaitu metode dengan implementasi yang lengkap. Metode ini dapat langsung dipanggil oleh objek kelas turunan.

Berikut ini adalah contoh implementasi

```python
from abc import ABC, abstractmethod

class Shape(ABC):
    @abstractmethod
    def area(self):
        pass

    @abstractmethod
    def perimeter(self):
        pass

class Rectangle(Shape):
    def __init__(self, length, width):
        self.length = length
        self.width = width

    def area(self):
        return self.length * self.width

    def perimeter(self):
        return 2 * (self.length + self.width)

rectangle = Rectangle(4, 5)
print(rectangle.area())      # Output: 20
print(rectangle.perimeter()) # Output: 18
```

B. Interface

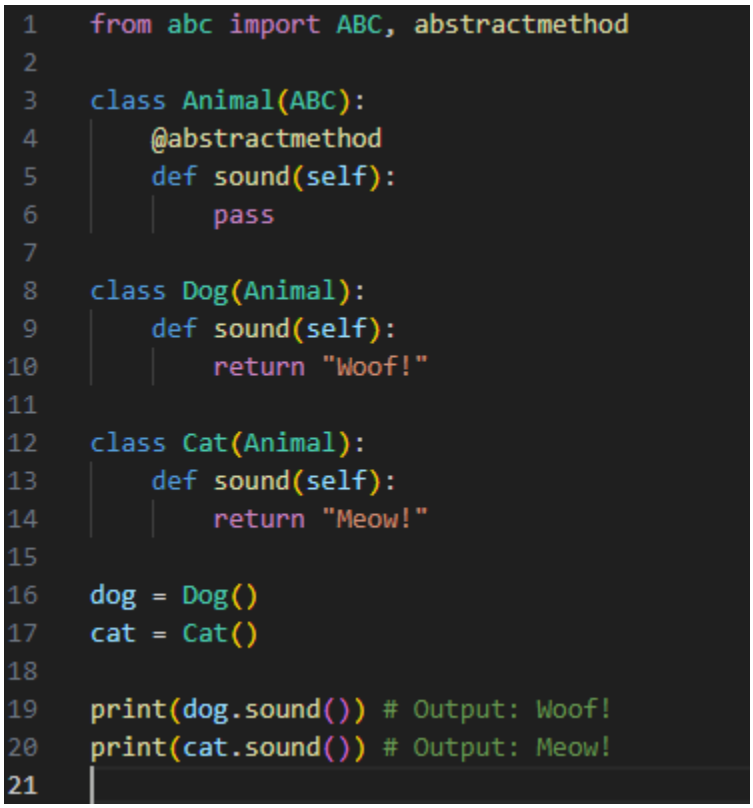
Interface adalah kontrak yang menentukan sekumpulan metode yang harus diimplementasikan oleh kelas-kelas yang menggunakan interface tersebut. Poin-poin penting tentang interface adalah:

1. Tidak memiliki implementasi: Interface hanya mendefinisikan metode-metode yang harus ada dalam kelas-kelas yang mengimplementasikannya. Interface tidak memiliki implementasi metode.

2. Kelas-kelas yang mengimplementasikan interface harus menyediakan implementasi metode yang didefinisikan dalam interface tersebut.

3. Satu kelas dapat mengimplementasikan beberapa interface.

Berikut ini adalah contoh implementasi kode sederhana yang menjelaskan interface:



```python
from abc import ABC, abstractmethod

class Animal(ABC):
    @abstractmethod
    def sound(self):
        pass

class Dog(Animal):
    def sound(self):
        return "Woof!"

class Cat(Animal):
    def sound(self):
        return "Meow!"

dog = Dog()
cat = Cat()

print(dog.sound()) # Output: Woof!
print(cat.sound()) # Output: Meow!
```

C. Metaclass

Metaclass adalah sebuah kelas yang mendefinisikan perilaku kelas. Dalam pemrograman berbasis objek, setiap objek adalah instance dari sebuah kelas, dan setiap kelas sendiri adalah instance dari sebuah metaclass. Metaclass memungkinkan kita untuk mengontrol bagaimana suatu kelas dibuat, dengan memberikan kemampuan untuk memodifikasi perilaku kelas, mengubah metode, atribut, dan lainnya. Dalam beberapa bahasa pemrograman seperti Python, metaclass dapat digunakan untuk mengubah perilaku dasar kelas, seperti memaksa adanya metode tertentu atau menambahkan fungsi tambahan ke semua kelas yang dibuat dari metaclass tersebut.Beberapa poin penting tentang metaclass adalah:

1. Metaclass adalah kelas dari kelas. Dengan kata lain, metaclass adalah pengontrol kelas.

2. Metaclass digunakan untuk memodifikasi perilaku kelas saat dibuat.

3. Salah satu penggunaan umum metaclass adalah untuk menerapkan pola desain seperti Singleton dan Factory.

Berikut ini adalah contoh implementasi kode sederhana yang menjelaskan metaclass:

```python
class Meta(type):
    def __new__(cls, name, bases, attrs):
        attrs['description'] = "This is a metaclass example."
        return super().__new__(cls, name, bases, attrs)

class MyClass(metaclass=Meta):
    pass

obj = MyClass()
print(obj.description)  # Output: This is a metaclass example.
```

Kesimpulan

Kelas abstrak digunakan sebagai kerangka dasar untuk kelas turunannya, menyediakan implementasi dasar untuk metode-metode yang akan diimplementasikan oleh kelas-kelas turunan. Interface, di sisi lain, adalah kontrak yang mengharuskan kelas yang mengimplementasikan interface tersebut untuk menyediakan implementasi metode yang didefinisikan dalam interface. Interface memungkinkan polimorfisme dan membantu dalam mencapai abstraksi yang lebih tinggi. Metaclass adalah kelas yang digunakan untuk membuat kelas baru dan mengendalikan perilaku kelas secara dinamis. Metaclass memberikan fleksibilitas dalam mengubah atau menambahkan fitur ke dalam kelas. Semua konsep ini merupakan bagian penting dari pemrograman berorientasi objek dan memberikan fleksibilitas dalam mengorganisir dan mengelola kode.

Apa itu interface dan kapan kita perlu memakainya?

- Interface adalah sebuah kontrak yang menyatakan metode-metode yang harus diimplementasikan oleh kelas yang mengimplementasikan interface tersebut. Interface tidak memiliki implementasi metode, hanya deklarasi metode tanpa detail implementasi. Interface digunakan ketika Anda ingin memastikan bahwa beberapa kelas memiliki perilaku yang serupa atau harus mengimplementasikan metode tertentu. Dalam situasi seperti ini, Anda dapat menggunakan interface sebagai panduan untuk memastikan bahwa kelas-kelas tersebut memiliki metode yang diperlukan.

Apa itu kelas abstrak dan kapan kita perlu memakainya? Apa perbedaannya dengan Interface?

- Kelas abstrak adalah kelas yang tidak dapat diinstansiasi dan berfungsi sebagai kerangka dasar untuk kelas turunannya. Kelas abstrak dapat memiliki metode abstrak (tanpa implementasi) dan metode konkrit (dengan implementasi). Kelas abstrak digunakan ketika Anda ingin memiliki beberapa perilaku umum yang dibagikan oleh kelas-kelas turunannya, tetapi Anda tidak ingin kelas abstrak tersebut diinstansiasi secara langsung.

Apa itu kelas konkret dan kapan kita perlu memakainya?

- Kelas konkret adalah kelas yang dapat diinstansiasi secara langsung. Kelas konkret adalah kelas biasa yang memiliki implementasi lengkap untuk semua metode dan dapat digunakan untuk membuat objek. Ketika Anda ingin membuat objek dari sebuah kelas dan menggunakan implementasi lengkap yang telah ditentukan, Anda perlu menggunakan kelas konkret.

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

- Metaclass adalah kelas yang digunakan untuk membuat kelas baru. Metaclass memungkinkan Anda mengendalikan pembuatan kelas, mengubah perilaku kelas, atau

menambahkan fungsi baru ke dalam kelas. Dengan menggunakan metaclass, Anda dapat memanipulasi kelas secara dinamis pada tingkat yang lebih tinggi.

- Kita perlu menggunakan metaclass ketika kita ingin mengontrol atau mengubah perilaku pembuatan kelas secara global. Misalnya, Anda dapat menggunakan metaclass untuk memodifikasi atribut atau metode yang ada dalam kelas sebelum kelas tersebut dibuat. Metaclass dapat digunakan untuk memodifikasi dan mempengaruhi kelas secara luas, sedangkan inheritance biasa hanya memungkinkan Anda mewarisi perilaku dari satu kelas ke kelas lain. Dengan metaclass, Anda dapat memanipulasi dan mengatur perilaku kelas secara dinamis sebelum pembuatan kelas terjadi.

Daftar Pustaka:

- Documentation Python: https://docs.python.org/3/library/abc.html
- Real Python - Abstract Base Classes in Python: https://realpython.com/python-abc-abstract-base-classes/
- Real Python - Python Class Inheritance: https://realpython.com/python-class-inheritance/
- Real Python - Metaclasses in Python: https://realpython.com/python-metaclasses/