LAPORAN PRAKTIKUM
PEMROGRAMAN BERORIENTASI OBJEK

Disusun Oleh :

Fauzan alfa abhista (121140217)

PROGRAM STUDI TEKNIK INFORMATIKA
INSTITUT TEKNOLOGI SUMATERA
2023

DAFTAR ISI

LAPORAN

Penjelasan singkat kelas abstrak, interface dan metakelas

Kelas abstrak, interface, dan metaclass adalah konsep penting dalam pemrograman berorientasi objek. Kelas abstrak adalah kelas yang tidak dapat diinstansiasi dan digunakan untuk mewakili konsep umum. Interface adalah kontrak yang menyatakan metode dan properti yang harus diimplementasikan oleh kelas lain. Metaclass adalah kelas yang digunakan untuk membuat kelas lainnya.

Kelas abstrak

Kelas Abstrak Kelas abstrak adalah kelas yang tidak dapat diinstansiasi. Kelas ini hanya digunakan untuk mewakili konsep umum dan menjadi tempat untuk mendefinisikan metode yang harus diimplementasikan oleh kelas anak. Untuk membuat kelas abstrak di Python, kita dapat menggunakan modul abc (Abstract Base Classes). Contoh implementasi kelas abstrak:

```
1. from abc import ABC, abstractmethod
2.
3. class Shape(ABC):
4.     @abstractmethod
5.     def area(self):
6.         pass
7.
8. class Rectangle(Shape):
9.     def __init__(self, width, height):
10.        self.width = width
11.        self.height = height
12.
13.    def area(self):
14.        return self.width * self.height
15.
16.r = Rectangle(10, 20)
17.print(r.area()) # Output: 200
```

Interface

Interface adalah kontrak yang menyatakan metode dan properti yang harus diimplementasikan oleh kelas lain. Di Python, tidak ada sintaksis untuk mendefinisikan interface secara eksplisit. Namun, kita dapat menggunakan kelas abstrak untuk mensimulasikan konsep interface. Contoh implementasi interface.

```
1. from abc import ABC, abstractmethod
2.
3. class Shape(ABC):
4.     @abstractmethod
```

```
    def area(self):
        pass

class Drawable(ABC):
    @abstractmethod
    def draw(self):
        pass

class Circle(Shape, Drawable):
    def __init__(self, radius):
        self.radius = radius

    def area(self):
        return 3.14 * (self.radius ** 2)

    def draw(self):
        print("Drawing circle...")

c = Circle(5)
print(c.area()) # Output: 78.5
c.draw() # Output: Drawing circle...
```

Metaclass

Metaclass adalah kelas yang digunakan untuk membuat kelas lainnya. Dalam Python, setiap kelas sebenarnya merupakan objek dari metaclass. Secara default, metaclass yang digunakan adalah type. Namun, kita dapat membuat metaclass sendiri dengan mendefinisikan kelas yang mewarisi dari type dan mengimplementasikan metode baru. Contoh implementasi metaclass:

```
class Singleton(type):
    _instances = {}

    def __call__(cls, *args, **kwargs):
        if cls not in cls._instances:
            cls._instances[cls] = super().__call__(*args, **kwargs)
        return cls._instances[cls]

class MyClass(metaclass=Singleton):
    pass

a = MyClass()
b = MyClass()
print(a is b) # Output: True
```

KESIMPULAN

Interface adalah sebuah kontrak yang menyatakan metode dan properti yang harus diimplementasikan oleh kelas lain. Interface tidak memiliki implementasi kode, melainkan hanya berisi definisi metode dan properti yang harus diimplementasikan oleh kelas lain. Konsep interface berguna untuk membuat kode lebih modular dan fleksibel, karena memungkinkan berbagai kelas yang berbeda untuk diatur secara homogen, meskipun mereka memiliki implementasi yang berbeda. Kita perlu menggunakan interface ketika kita ingin membuat kelas-kelas yang berbeda, tetapi memiliki metode dan properti yang sama. Contohnya, jika kita ingin membuat beberapa kelas geometri seperti Lingkaran, Segitiga, dan Persegi Panjang, kita dapat membuat sebuah interface Shape yang mendefinisikan metode area(), sehingga setiap kelas geometri yang kita buat harus mengimplementasikan metode area(). Hal ini akan memudahkan kita dalam membuat kode yang mudah dipelihara dan diubah di masa depan, karena setiap kelas geometri memiliki metode yang sama dan dapat dipanggil secara homogen.

Kelas abstrak adalah sebuah kelas yang tidak dapat diinstansiasi dan memiliki setidaknya satu metode abstrak. Metode abstrak adalah metode yang tidak memiliki implementasi dan hanya berisi definisi metode yang harus diimplementasikan oleh kelas anak. Konsep kelas abstrak berguna untuk memaksakan implementasi metode yang sama pada kelas-kelas anak, sehingga memudahkan kita dalam membuat kode yang konsisten dan mudah dipelihara.

Perbedaan antara kelas abstrak dan interface adalah pada implementasinya. Interface hanya menyatakan metode dan properti yang harus diimplementasikan oleh kelas lain, tanpa memiliki implementasi kode sedikitpun. Sementara itu, kelas abstrak dapat memiliki implementasi kode, tetapi juga dapat memiliki metode abstrak yang harus diimplementasikan oleh kelas anak.

Kelas konkret (concrete class) adalah sebuah kelas yang dapat diinstansiasi dan memiliki implementasi lengkap dari setiap metode yang dimilikinya. Dalam bahasa pemrograman Python, hampir semua kelas yang kita definisikan adalah kelas konkret. Kita perlu menggunakan kelas konkret ketika kita ingin membuat sebuah objek yang memiliki implementasi lengkap dari setiap metode yang dimilikinya. Contohnya, jika kita ingin membuat sebuah objek bernama Person, kita dapat membuat sebuah kelas konkret bernama Person yang memiliki implementasi lengkap dari setiap metodenya seperti getName(), getAge(), setAddress(), dll.

Namun, kita juga dapat menggunakan konsep pewarisan (inheritance) untuk membuat kelas konkret baru yang memiliki metode yang sama dengan kelas parentnya. Kita dapat membuat sebuah kelas konkret baru yang mewarisi semua metode dan properti dari kelas parentnya, dan kemudian menambahkan metode dan properti baru yang diperlukan oleh kelas tersebut. Sebagai contoh, jika kita ingin membuat sebuah kelas untuk merepresentasikan seekor kucing, kita dapat membuat sebuah kelas konkret bernama Cat yang mewarisi semua metode dan properti dari kelas Animal (kelas parent), seperti eat(), sleep(), dan breed(). Kemudian, kita dapat menambahkan metode dan properti baru yang spesifik untuk kucing, seperti meow() dan fur_color().

Metaclass adalah sebuah kelas yang digunakan untuk membuat kelas baru. Dalam bahasa pemrograman Python, setiap kelas memiliki sebuah metaclass yang menentukan bagaimana kelas tersebut akan dibuat dan bekerja. Kita perlu menggunakan metaclass ketika kita ingin mengontrol cara

sebuah kelas dibuat dan beroperasi.

Dalam Python, kita dapat membuat metaclass dengan mengimplementasikan metode-metode tertentu dalam sebuah kelas. Dalam metaclass, kita dapat mengimplementasikan metode seperti new() yang akan dipanggil ketika sebuah kelas baru dibuat, atau init() yang akan dipanggil ketika sebuah objek baru dari kelas tersebut dibuat. Kita juga dapat mengimplementasikan metode lainnya seperti call() untuk mengatur perilaku kelas saat diinstansiasi, atau getitem() untuk mengatur perilaku kelas saat diakses melalui indeks.

Perbedaan utama antara metaclass dan inheritance biasa adalah pada saat pembuatan kelas. Dalam inheritance biasa, sebuah kelas dibuat dengan mewarisi metode dan properti dari kelas parent. Sementara itu, dalam metaclass, sebuah kelas dibuat dengan menggunakan sebuah kelas khusus yang bertindak sebagai metaclass.

DAFTAR PUSTAKA

Beazley, D. M. (2019). *Python Essential Reference (5th Edition).* Addison-Wesley Professional.

Downey, A. B. (2015). *Think Python: How to Think Like a Computer Scientist (2nd Edition).* O'Reilly Media.

Lutz, M. (2013). *Learning Python (5th Edition).* O'Reilly Media.

Pilgrim, M. (2009). *Dive Into Python 3.* Apress.

Rossum, G. V. (2009). *Python 3 Reference Manual.* CreateSpace.

van Rossum, G. &. (2001). *Python 2.1.* Bible.