TUGAS MODUL 6
PEMROGRAMAN BERBASIS OBJEK

Disusun Oleh :

Bima Setiawan Sandi

121140162

PROGRAM STUDI TEKNIK INFORMATIKA

INSTITUT TEKNOLOGI SUMATERA

2023

DAFTAR ISI

KELAS ABSTRAK

Kelas Abstrak dapat dianggap sebagai garis besar untuk kelas yang berbeda. Ini memungkinkan untuk menggambarkan banyak teknik yang harus dikerjakan di dalam kelas anak mana pun yang dikerjakan dari kelas abstrak. Kelas abstrak adalah kelas yang memiliki setidaknya satu metode abstrak. Sebuah metode tanpa implementasi dikenal sebagai metode abstrak. Kelas abstrak digunakan untuk merancang unit fungsional yang besar. Kelas abstrak digunakan untuk menyediakan antarmuka umum untuk berbagai implementasi komponen.

Python tidak datang dengan kelas abstrak secara default. ABC adalah nama modul Python yang berfungsi sebagai dasar untuk mendefinisikan kelas-kelas Basis Abstrak (ABC). Fungsi ABC dengan mendaftarkan kelas beton sebagai implementasi dari basis abstrak dan mendekorasi kelas dasar sebagai metode abstrak. Ketika kata kunci @abstractmethod digunakan untuk menghias sebuah metode, itu membuatnya lebih abstrak.

Contoh Implementasi kode sederhana Kelas Abstrak :

```
1. from abc import ABC, abstractmethod
2.
3. class Animal(ABC):
4.
5.     @abstractmethod
6.     def sound(self):
7.         pass
8.
9. class Cat(Animal):
10.
11.    def sound(self):
12.        print("Meong")
13.
14.class  Dog(Animal):
15.
16.    def sound(self):
17.        print("Guk Guk")
18.
19.cat = Cat()
20.cat.sound() # Output: Meong
21.
22.dog = Dog()
23.dog.sound()  # Output: Guk Guk
24.
```

INTERFACE

Dalam Python, interface adalah kerangka kerja dasar atau blueprint yang digunakan untuk mengetahui metode atau fungsi mana yang harus ada di kelas tanpa harus mengimplementasikannya secara mendetail. Di Python, kelas abstrak atau kelas abstrak biasanya digunakan untuk mendefinisikan interface. Interface berfungsi dengan kemajuan pemrograman dalam kelompok, mengurangi bug dan kesalahan, meningkatkan kemampuan beradaptasi, dan mempertimbangkan polimorfisme. Meskipun demikian, Python sebenarnya tidak memiliki interface yang jelas seperti bahasa pemrograman lain. Namun, tujuan yang sama dapat dicapai dengan penggunaan kelas abstrak seperti antarmuka dalam bahasa pemrograman lain.

Contoh implementasi kode sederhana Interface :

```
1. from abc import ABC, abstractmethod
2.
3. class Shape(ABC):
4.     @abstractmethod
5.     def luas(self):
6.         pass
7.
8.     @abstractmethod
9.     def keliling(self):
10.        pass
11.
12.class PersegiPanjang(Shape):
13.    def __init__(self, lebar, panjang):
14.        self.lebar = lebar
15.        self.panjang = panjang
16.
17.    def luas(self):
18.        return self.lebar * self.panjang
19.
20.    def keliling(self):
21.        return 2 * (self.lebar + self.panjang)
22.
23.class Lingkaran(Shape):
24.    def __init__(self, jarijari):
25.        self.jarijari = jarijari
26.
27.    def luas(self):
28.        return 3.14 * self.jarijari ** 2
29.
30.    def keliling(self):
31.        return 2 * 3.14 * self.jarijari
32.
33.# Membuat objek PersegiPanjang
34.PersPanj = PersegiPanjang(5, 10)
35.print("Luas Persegi Panjang: ", PersPanj.luas())
```

```
print("Keliling Persegi Panjang: ", PersPanj.keliling())

# Membuat objek Lingkaran
lingkaran = Lingkaran(7)
print("Luas Lingkaran: ", lingkaran.luas())
print("Keliling Lingkaran: ", lingkaran.keliling())

```

METACLASS

Di Python, kelas yang disebut metaclass digunakan untuk membuat kelas baru. Metaclass biasanya adalah kelas dengan kontrol atas objek kelas, seperti menambahkan atribut dan metode kelas baru atau mengubah perilaku kelas yang ada.

Setiap kelas di Python adalah keturunan tidak langsung dari metaclass, biasanya metaclass bawaan yang dikenal sebagai "type". Meskipun demikian, Anda dapat membuat metaclass kustom Anda sendiri dengan memperoleh kelas dari sejenis.

Saat membuat objek kelas baru atau memvalidasi objek kelas yang sudah ada, penggunaan metaclass dapat membantu. Namun, sebagian besar pengembangan perangkat lunak Python tidak memerlukan metaclass karena dapat membuat kode menjadi lebih rumit dan sulit dibaca.

Contoh implementasi kode sederhana metaclass :

```
1. class MyMeta(type):
2.     def __new__(cls, name, bases, attrs):
3.         new_attrs = {}
4.         for attr_name, attr_value in attrs.items():
5.             if not callable(attr_value):
6.                 new_attrs[attr_name.upper()] = attr_value
7.             else:
8.                 new_attrs[attr_name] = attr_value
9.         return super().__new__(cls, name, bases, new_attrs)
10.
11.class MyClass(metaclass=MyMeta):
12.x = 10
13.     def say_hello(self):
14.         print("Hello, World!")
15.
```

KESIMPULAN

Interface adalah kerangka kerja dasar atau blueprint yang digunakan untuk mengetahui metode atau fungsi mana yang harus ada di kelas tanpa harus mengimplementasikannya secara mendetail. Di Python, kelas abstrak atau kelas abstrak biasanya digunakan untuk mendefinisikan interface.

Kelas abstrak adalah kelas yang memiliki setidaknya satu metode abstrak. Sebuah metode tanpa implementasi dikenal sebagai metode abstrak. Kelas abstrak digunakan untuk merancang unit fungsional yang besar. Kelas abstrak digunakan untuk menyediakan antarmuka umum untuk berbagai implementasi komponen. kelas abstrak lebih fleksibel daripada interface karena memungkinkan metode yang diimplementasikan dan tidak diimplementasikan, sementara interface hanya memungkinkan metode yang tidak diimplementasikan.

Kelas konkret (concrete class) adalah jenis kelas yang dapat langsung diinstansiasi objeknya, berbeda dengan kelas abstrak dan interface yang tidak dapat diinstansiasi secara langsung. Kelas konkret merupakan implementasi konkret dari kelas abstrak atau interface, yang telah ditentukan cara kerja dan perilaku objeknya. Kita perlu menggunakan kelas konkret ketika kita ingin membuat objek yang bersifat konkret atau spesifik, dan sudah memiliki implementasi lengkap dari atribut dan metodenya.

Metaclass adalah sebuah kelas yang digunakan untuk membuat kelas baru. Sebuah metaclass mengontrol cara suatu kelas dibangun dengan mengatur atribut dan perilaku dari kelas tersebut. Dalam Python, metaclass digunakan untuk membuat struktur kelas secara dinamis pada waktu eksekusi. Kita perlu menggunakan metaclass ketika ingin membuat kelas baru dengan cara yang berbeda dari kelas konvensional. Perbedaan utama antara metaclass dengan inheritance biasa adalah bahwa inheritance mengizinkan pewarisan atribut dan perilaku dari kelas dasar ke kelas turunan, sedangkan metaclass mengizinkan pengaturan atribut dan perilaku untuk pembuatan kelas baru.

DAFTAR PUSTAKA

[1] A. Muhardian, "Tutorial Java OOP: Mengenal Class Abstrak dan Cara Pakainya," Petani Kode, 5 Januari 2020. [Online]. Available: https://www.petanikode.com/java-oop-abstract/. [Accessed 11 April 2023].

[2] Geeks For Geeks, "Abstract Classes in Python," GeeksforGeeks, 19 Maret 2021. [Online]. Available: https://www.geeksforgeeks.org/abstract-classes-in-python/. [Accessed 11 April 2023].

[3] W. Murphy, "Implementing an Interface in Python," Realpython, 10 Februari 2020. [Online]. Available: https://realpython.com/python-interface/. [Accessed 12 April 2023].

[4] J. Sturtz, "Python Metaclasses," Realpython, 15 Mei 2018. [Online]. Available: https://realpython.com/python-metaclasses/. [Accessed 12 April 2023].

[5] N. Huda, "Python: Kelas Dan Objek," Jagongoding, 13 Maret 2021. [Online]. Available: https://jagongoding.com/python/menengah/oop/kelas-dan-objek/. [Accessed 12 April 2023].