RESUME MATERI

TUGAS 6

M MALIK AGUSTIAN

121140148

RB

PROGRAM STUDI TEKNIK INFORMATIKA

JURUSAN TEKNOLOGI PRODUKSI DAN INDUSTRI

INSTITUT TEKNOLOGI SUMATERA

2023

DAFTAR ISI

KELAS ABSTRAK

Dalam Bahasa python, kelas abstrak merupakan kelas yang tidak dapat berdiri sendiri dan dipakai sendiri, melainkan sebagai struktur dasar yang akan mewarisi kelas yang dituju (kelas turunan). Kelas abstrak mempunyai keyword *abstractmethod(ABC)*. Abstract method sendiri merupakan method yang akan di implementasikan ke child class.

Kelas abstrak umumnya bersifat memaksa method yang memakai kelas tersebut untuk memakai isi dari kelas abstrak, pemakaiannya bisa hanya memakai logika atau mewarisi secara penuh kelas abstrak tersebut. Kelas abstrak berhubungan erat dengan pewarisan (inheritance).

Pemanggilan kelas abstrak dapat dilakukan dengan potongan kode sebagai berikut

from abc import ABC, abstractmethod

```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, width, height):
        self.width = width
        self.height = height

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

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

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

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

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

# Menggunakan objek dari kelas abstrak tidak diizinkan
# shape = Shape()  # Ini akan menghasilkan error

# Membuat objek dari kelas turunan
rectangle = Rectangle(5, 3)
circle = Circle(4)

# Memanggil metode pada objek-objek tersebut
print("Luas persegi panjang:", rectangle.area())  # Output: 15
print("Keliling persegi panjang:", rectangle.perimeter())  # Output: 16
print("Luas lingkaran:", circle.area())  # Output: 50.24
print("Keliling lingkaran:", circle.perimeter())  # Output: 25.12
```

Pada contoh di atas, Shape adalah kelas abstrak yang memiliki dua metode abstrak: area() dan perimeter(). Kelas Rectangle dan Circle adalah kelas turunan yang mewarisi dari kelas Shape dan mengimplementasikan metode-metode yang diwarisi.

Karena Shape adalah kelas abstrak, kita tidak dapat membuat objek langsung dari kelas tersebut. Namun, kita dapat membuat objek dari kelas turunan seperti Rectangle dan Circle.

INTERFACE

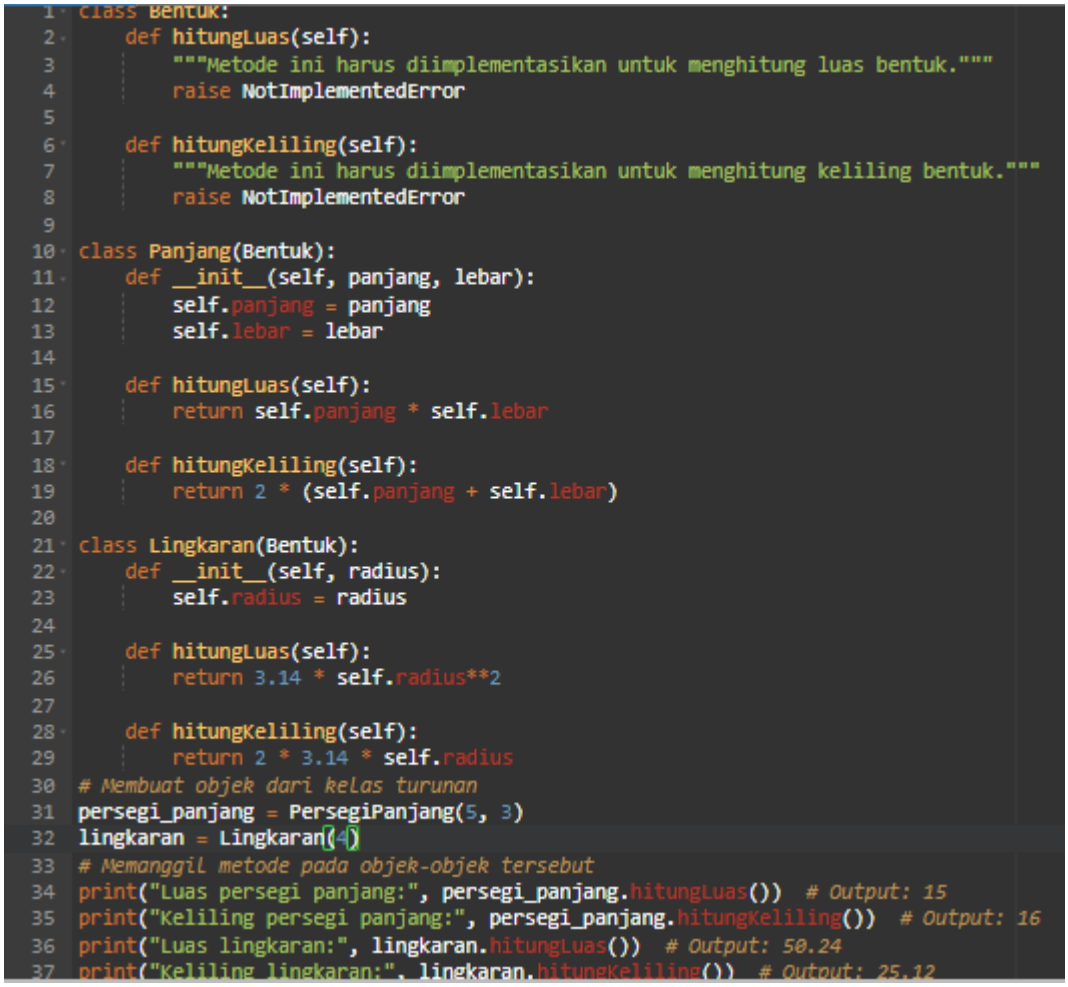
Interface adalah kontrak atau spesifikasi yang mendefinisikan metode atau perilaku yang harus diimplementasikan oleh kelas-kelas lain. Ini memungkinkan pemisahan antara antarmuka publik dan implementasi privat dari sebuah kelas.

Berikut adalah beberapa pendekatan yang dapat digunakan untuk mencapai konsep "interface" dalam Python:

1. Pemisahan antarmuka dan implementasi dapat mendefinisikan sebuah kelas dengan metode-metode yang perlu diimplementasikan, tetapi tidak memberikan implementasi yang sebenarnya. Kelas-kelas lain kemudian dapat mengimplementasikan metode-metode tersebut sesuai dengan kebutuhan.
2. Penggunaan kelas abstrak dapat menggunakan modul abc (Abstract Base Classes) yang disediakan oleh Python untuk mendefinisikan kelas abstrak. Kelas abstrak dapat berperan sebagai "interface" dengan memperkenalkan metode-metode yang harus diimplementasikan oleh kelas turunannya.

Ada 2 jenis dari interface yaitu informal dan formal interface.

a. Informal Interfaces, merupakan aturan yang tidak terlalu memaksakan, seperti class nya dapat di override.



```
class Bentuk:
    def hitungLuas(self):
        """Metode ini harus diimplementasikan untuk menghitung luas bentuk."""
        raise NotImplementedError

    def hitungKeliling(self):
        """Metode ini harus diimplementasikan untuk menghitung keliling bentuk."""
        raise NotImplementedError

class Panjang(Bentuk):
    def __init__(self, panjang, lebar):
        self.panjang = panjang
        self.lebar = lebar

    def hitungLuas(self):
        return self.panjang * self.lebar

    def hitungKeliling(self):
        return 2 * (self.panjang + self.lebar)

class Lingkaran(Bentuk):
    def __init__(self, radius):
        self.radius = radius

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

    def hitungKeliling(self):
        return 2 * 3.14 * self.radius
# Membuat objek dari kelas turunan
persegi_panjang = PersegiPanjang(5, 3)
lingkaran = Lingkaran(4)
# Memanggil metode pada objek-objek tersebut
print("Luas persegi panjang:", persegi_panjang.hitungLuas())  # Output: 15
print("Keliling persegi panjang:", persegi_panjang.hitungKeliling())  # Output: 16
print("Luas lingkaran:", lingkaran.hitungLuas())  # Output: 50.24
print("Keliling lingkaran:", lingkaran.hitungKeliling())  # Output: 25.12
```

b. Formal Interfaces, merupakan jenis interface yang aturannya ketat berdasarkan abc module.

Gambar diatas merupakan contoh penerapan interface.

Terdapat beberapa perbedaan interface dan abstraksi. Perbedaannya yaitu sebagai berikut

1. Pada kelas abstrak dimungkinkan membuat properti atau variabel sedangkan di interface hanya bisa buat konstanta saja
2. Pada kelas abstrak bisa implementasikan kode method seperti class biasa, sedangkan di interface harus menggunakan default
3. Pada kelas abstrak dapat memiliki member private dan protected sedangkan interface harus public
4. Class abstrak diimpelentasikan dengan pewarisan (extends) sedangkan interaface menggunakan implements

METACLASS

Metaclass di python merupakan kelas yang dapat membuat kelas lain, metaclass dapat merubah perilaku atau atribut suatu kelas. Metaclass adalah objek dari *type*. Metaclass dapat dibuat dengan cara membuat acuan type di Kelas yang dituju, contohnya adalah

Class MetaClass(type)

Pada dasarnya, metaclass adalah "kelas dari kelas" atau "blueprint" yang digunakan untuk membuat kelas-kelas. Ketika mendefinisikan sebuah kelas, Python menggunakan metaclass untuk membuat objek kelas tersebut. Metaclass dapat mempengaruhi atribut-atribut, metode, dan perilaku lainnya dari kelas yang dihasilkan.

Metaclass memiliki 2 versi yaitu

a. Old-Style Classes

Dengan kelas gaya lama, kelas dan tipe bukanlah hal yang persis sama. Instance dari kelas gaya lama selalu diimplementasikan dari satu tipe bawaan yang disebut instance. If objadalah turunan dari kelas gaya lama, obj.__class__tentukan kelasnya, tetapi type(obj)selalu instance

```
Python >>>

>>> class Foo:
...     pass
...
>>> x = Foo()
>>> x.__class__
<class __main__.Foo at 0x000000000535CC48>
>>> type(x)
<type 'instance'>
```

b. New Style Classes

Kelas gaya baru menyatukan konsep kelas dan tipe. Jika objmerupakan turunan dari kelas gaya baru, type(obj)sama dengan obj._class_:

```
Python >>>

>>> class Foo:
...     pass
>>> obj = Foo()
>>> obj.__class__
<class '__main__.Foo'>
>>> type(obj)
<class '__main__.Foo'>
>>> obj.__class__ is type(obj)
True
```

DAFTAR PUSTAKA

https://towardsdatascience.com/how-to-use-abstract-classes-in-python-d4d2ddc02e90

https://www.geeksforgeeks.org/python-metaclasses/

https://realpython.com/python-interface/

https://www.geeksforgeeks.org/python-interface-module/

https://realpython.com/python-metaclasses/