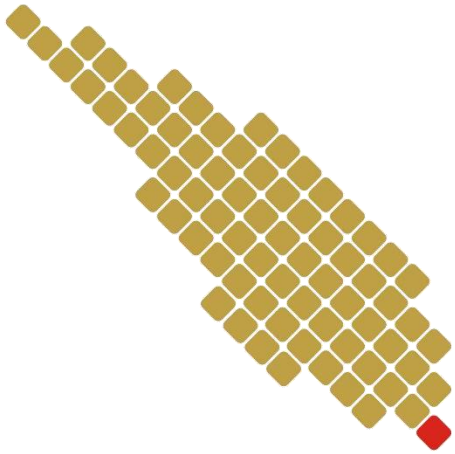
TUGAS PRAKTIKUM PBO MINGGU 6

PENGERTIAN SERTA PENERAPAN DARI

KONSEP ABSTRACT CLASS, INTERFACE, DAN METACLASS



Disusun oleh:

Nama : Fathurrahman Al Hafid
NIM : 121140088
Kelas : Praktikum PBO (RB)

PROGRAM STUDI TEKNIK INFORMATIKA

INSTITUT TEKNOLOGI SUMATERA LAMPUNG SELATAN

2023

A. Kelas Abstrak

Kelas Abstrak adalah kelas yang bukan objek nyata dan digunakan sebagai model untuk kelas lain. Umumnya, Kelas Abstrak akan memiliki metode abstrak. Metode Abstrak adalah metode dasar yang harus diimplementasikan kembali di kelas anak. Metode abstrak ditulis tanpa isi metode, tetapi hanya 'tanda tangan'. Tanda tangan suatu metode adalah bagian dari metode yang terdiri dari nama metode dan parameternya (jika ada) [1]. Kelas abstrak adalah jenis kelas yang digunakan untuk membuat struktur pewarisan. Ini berarti bahwa semua kelas yang diturunkan dari kelas abstrak akan memiliki kumpulan metode yang sama, apa pun jenis objek spesifik yang diwakili oleh kelas tersebut. Kelas abstrak umumnya digunakan dalam pemrograman berorientasi objek untuk membuat struktur logis untuk pewarisan kelas [2]. Penggunaan absrak kelas ditandai dengan menggunakan sebuah destructor yaitu *@abstractmethod* di dalam class serta import modul abc bawaan python pada awal program.

Contoh codingan:

```
from abc import ABC, abstractmethod #Proses Import Modul abc bawaan Python    "Modul": Unknown

class BangunDatar(ABC):    "Bangun": Unknown word.
    @abstractmethod #Penggunaan destructor    "Penggunaan": Unknown word.
    def hitung_luas(self): #inisialisasi class parent, namun di dalam class diisi dengan pass
        pass

class Lingkaran(BangunDatar): #Inisiasi class child berdasarkan class parent diatas    "Lingka
    def __init__(self, r):
        self.r = r

    def hitung_luas(self): #Method parent yang harus diambil oleh class childe    "hitung": Un
        return 3.14 * self.r ** 2

class Persegi(BangunDatar):    "Persegi": Unknown word.
    def __init__(self, sisi):    "sisi": Unknown word.
        self.sisi = sisi    "sisi": Unknown word.

    def hitung_luas(self):    "hitung": Unknown word.
        return self.sisi ** 2    "sisi": Unknown word.
```

B. Interface

Interface Objek adalah seperangkat aturan atau harapan yang disetujui oleh objek ('klien') dan pemasok objek ('server'). Ini memudahkan kedua belah pihak untuk bekerja sama karena mereka tahu apa yang diharapkan dari satu sama lain.

Jika Anda ingin menggunakan objek Interface, Anda harus memastikan bahwa kelas Anda mengimplementasikan semua metode di Interface. Ini sama dengan apa yang akan Anda lakukan jika Anda memprogram dalam bahasa berorientasi objek. Interface seperti kelas abstrak, tetapi digunakan untuk mendeskripsikan fitur atau kemampuan sesuatu yang lain. Interface seperti kelas abstrak, tetapi hanya berisi tanda tangan metode. Isi dari method tersebut akan dibuat ulang di dalam class yang menggunakan interface tersebut. Interface adalah seperangkat aturan yang harus diikuti kelas agar dapat menggunakan metode tertentu. Kelas harus dapat menemukan implementasi metode Interface untuk menggunakannya [3]. Interface adalah seperangkat aturan yang dapat disetujui oleh satu atau lebih kelas untuk diikuti. Setelah kelas setuju untuk mengikuti aturan Interface, ia dapat menggunakan metode dan konstanta Interface tanpa harus khawatir tentang kelas mana yang membuat Interface atau cara kerjanya. [4]. Interface biasanya digunakan untuk membuat fitur tertentu pada sebuah perangkat.

Contoh codingan:

```python
from abc import ABC, abstractmethod #Import modul abc dari bawaan Python    "modul":

class InterfaceKalkulator(ABC): #class turunnan dari ABC    "Kalkulator": Unknown wor
    @abstractmethod #decorator yang digunakan    "digunakan": Unknown word.
    def tambah(self, a, b): #fungsi pada class parent    "tambah": Unknown word.
        pass

    @abstractmethod
    def kurang(self, a, b): #fungsi pada class parent    "kurang": Unknown word.
        pass

class Kalkulator(InterfaceKalkulator): #class child turunan dari class parent    "Kal
    def tambah(self, a, b): #pengambilan method yang harus diambil oleh clas child
        return a + b

    def kurang(self, a, b): #pengambilan method yang harus diambil oleh clas child
        return a - b
```

C. Metaclass

Metaprogramming atau metaclass adalah cara suatu program untuk mempelajari dan memanipulasi kodenya sendiri. Dalam Python, metaclasses adalah jenis khusus dari kelas yang memungkinkan sebuah program untuk membuat aturan sendiri untuk bagaimana kode berperilaku.

Metaclass adalah jenis objek khusus yang Anda gunakan untuk membuat jenis objek baru. Anda tidak perlu terlalu sering khawatir tentang metaclass,

tetapi jika Anda perlu menggunakannya, umumnya mudah digunakan. Sebagian besar pemrogram Python tidak terlalu sering menggunakan metaclass, tetapi jika mereka melakukannya, mereka biasanya merasa mudah menggunakannya [5]. Metaclass merupakan tingkatan tertinggi dari hirearki class dalam python dimana class class tersebut merupakan turunan dari metaclass itu sendiri. Contoh dari metaclass adalah int float yang pada dasar nya di c++ atau java merupakan sebuah tipe data.namun pada python int dan float juga merupakan metaclass karena int dan float tersebut dapat diturunkan menjadi sebuah class baru.

Contoh codingan:

```
class MyMetaClass(type): #class yang diturunkan dari metaclass type    "diturunkan":
    def __new__(cls, name, bases, attrs): #method yang digunakan dalam class myclass
        # modifikasi atribut di kelas yang baru dibuat    "modifikasi": Unknown word.
        attrs['x'] = 1 #proses modifikasi atribut x    "modifikasi": Unknown word.
        attrs['y'] = 2 #proses modifikasi atribut y    "modifikasi": Unknown word.

        # panggil konstruktor superclass    "panggil": Unknown word.
        return super().__new__(cls, name, bases, attrs)

class MyClass(metaclass=MyMetaClass): #class turunan dari class mymetaclass    "turur
    def __init__(self, z):
        self.z = z

my_object = MyClass(3) #pemanggilan  objek dari class myclass    "pemanggilan": Unkno
print(my_object.x)  # output: 1
print(my_object.y)  # output: 2
print(my_object.z)  # output: 3
```

D. Kesimpulan

a. Apa itu interface dan kapan kita perlu memakainya?

Interface merupakan salah satu pengaplikasian class pada python dimana seluruh method yang berada di dalam class parent harus diimplementasikan semuanya ke dalam class child. Konsep interface digunakan saat ada sebuah class yang wajib menggunakan seluruh method dari parent. Konsep interface digunakan jika method dari class parent besifat konkret. Contoh nya jika kita membuat sebuah class parent yaitu fitur yang berisi method bluetotth dan wifi, maka saat kita membuat kelas turunannya misalkan oppo, maka class opp tersebut wajib mengimplementasikan method bluetototh serta wifi ke dalam class nya.

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

Abstrak class merupakan sebuah class parent dimana kita tidak bisa membuat objek secara langsung pada class tersebut karena bentuk

nya yang abstrak. Sehingga kita memerlukan class turunan untuk memanggil objek supaya dapat menggunakan method pada class parent nya. Class abstract digunakan ketika kita ingin menentukan kerangka atau template umum yang harus diikuti oleh kelas-kelas turunan, tetapi tidak ingin mengimplementasikan detail-detailnya di kelas abstrak itu sendiri. Contoh kelas abstrak adalah kelas "BangunDatar" yang memiliki metode abstrak "hitung luas" serta "hitung keliling" yang diimplementasikan pada class child seperti class persegi serta class segitiga.

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

Kelas konkret dalam pemrograman adalah kelas yang dapat digunakan untuk membuat objek baru. Artinya, Anda bisa membuat instance kelas ini dengan memanggil konstruktor kelas. Contoh kelas beton adalah motor. Kita bisa membuat objek baru sekelas ini seperti motor Honda atau motor YamahaSaat kita ingin membuat objek baru yang berbeda, kita perlu menggunakan kelas konkret. Misalnya, jika kita ingin membuat aplikasi manajemen restoran, kita dapat membuat kelas konkret seperti "MenuFood", "OrderFood", "Customer", dll, karena setiap objek yang kita buat dari kelas tersebut akan memiliki atribut dan metode yang berbeda. Kelas dapat digunakan untuk membuat kelas turunan atau subkelas, yang dapat berguna untuk membuat kelas dengan atribut dan metode yang spesifik dan terfokus. Menggunakan kelas konkret sebagai superclass dapat membuat proses ini lebih mudah.

Daftar Pustaka

[1] Python. [Online]. Available: https://docs.python.org/3/library/abc.html. [Accessed 09 April 2023].

[2] Andre, "duniailkom.com," 14 Oktober 2014. [Online]. Available: https://www.duniailkom.com/tutorial-belajar-oop-php-pengertian-abstract-class-dan-abstract-method-php/. [Accessed 09 April 2023].

[3] Andre, "duniailkom.com," 10 Oktober 2014. [Online]. Available: https://www.duniailkom.com/tutorial-belajar-oop-php-pengertian-object-interface-dalam-pemrograman-berbasis-objek/. [Accessed 09 April 2023].

[4] M. Zaelani, "segalahal.com," 19 Maret 2020. [Online]. Available: https://segalahal.com/2020/03/19/oop-java-interface-part-7/. [Accessed 09 April 2023].

[5] J. Sturtz, "realpython.com," 2013. [Online]. Available: https://realpython.com/python-metaclasses/. [Accessed 09 April 2023].