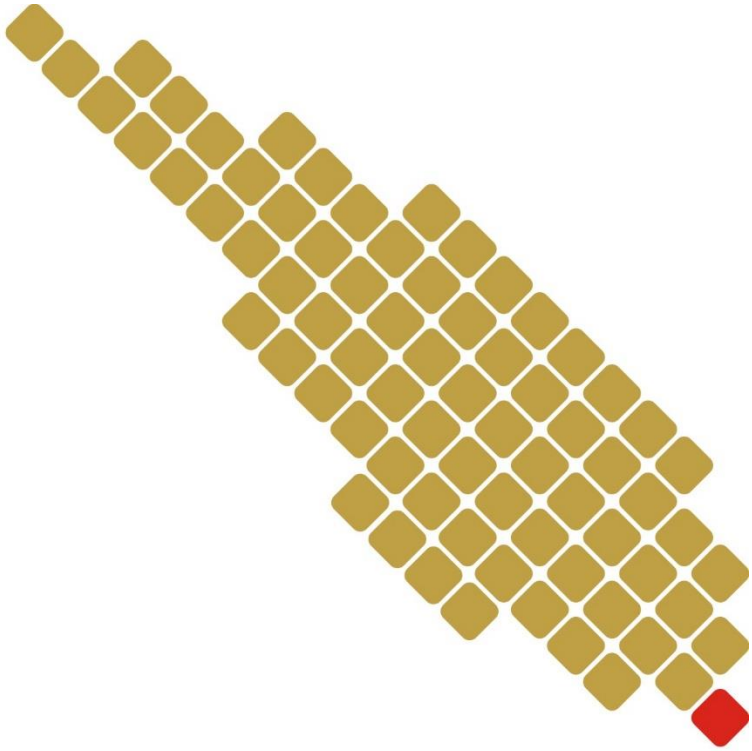
RESUME MATERI PRAKTIKUM PBO
MODUL 5

Nama : Juan Verrel Tanuwijaya
NIM : 121140072

Kelas : RB



ITERA

I. Kelas Abstrak

Abstract class dalam Python adalah kelas yang tidak dapat diinisiasi objeknya secara langsung. Sebaliknya, abstract class digunakan sebagai kerangka dasar untuk kelas turunan yang akan diimplementasikan dengan lebih spesifik dan lebih detail. Abstract class memiliki setidaknya satu metode abstrak yang harus diimplementasikan oleh kelas turunan. Metode abstrak hanya memiliki deklarasi tanpa implementasi, sehingga kelas turunan harus memberikan implementasi sesuai dengan kebutuhan.

Untuk membuat abstract class, kita harus menggunakan modul abc (Abstract Base Class) yang disediakan oleh Python, dengan cara menurunkan sebuah kelas dengan ‘ABC’. Kita juga dapat menandai metode yang ingin dibuat abstrak dengan menggunakan decorator @abstractmethod. Hal ini memaksa kelas turunan untuk memberikan implementasi pada metode tersebut sebelum dapat digunakan. Abstract class sangat berguna untuk memudahkan pengembangan aplikasi dan memastikan konsistensi antara kelas turunan yang mungkin dikembangkan oleh beberapa programmer.

Dalam penggunaannya, abstract class digunakan sebagai kerangka dasar untuk kelas turunan yang lebih spesifik. Dengan membuat abstract class dan memaksa kelas turunan untuk mengimplementasikan metode tertentu, kita dapat memastikan bahwa kelas turunan yang berbeda-beda akan memiliki kesamaan dalam perilaku dan memastikan konsistensi dalam pengembangan aplikasi.

```
1.  class Hewan(ABC):
2.      @abstractmethod
3.      def Rumah(self):
4.          pass
5.
6.  class Peliharaan(Hewan):
7.      def Rumah(self):
8.          print("Didalam rumah pemilik hewan")
9.
10. class Liar(Hewan):
11.     def Rumah(self):
12.         print("Di alam rimba")
```

Dapat dilihat telah dibuat sebuah Class Hewan yang diturunkan ‘ABC’ sehingga membuatnya menjadi sebuah abstract class, dikarenakan Hewan merupakan abstract class akan terjadi error jika kita menginisiasi objek berdasarkan class hewan.

```
13. kangguru=Hewan()
```

```
Exception has occurred: TypeError ×
Can't instantiate abstract class Hewan with abstract method Rumah
  File "D:\Juan\belajar\PBO\implementasi.py", line 16, in <module>
    kangguru=Hewan()
             ^^^^^^^
TypeError: Can't instantiate abstract class Hewan with abstract method Rumah
```

Dan dikarenakan method Rumah merupakan abstract method, setiap keturunan dari kelas Hewan harus memiliki method Rumah jika tidak akan terjadi error pada program.

II. Interface

Interface dalam Python adalah konsep yang terkait dengan abstraksi dan polimorfisme. Interface adalah kumpulan metode dan konstanta yang didefinisikan dalam sebuah kelas, tetapi tidak memiliki implementasi aktual. Oleh karena itu, implementasi kelas yang menggunakan interface harus menyediakan implementasi untuk semua metode dan konstanta yang didefinisikan dalam interface.

Dalam Python, interface biasanya didefinisikan menggunakan kelas abstrak. Kelas abstrak adalah kelas yang tidak dapat diinstansiasi dan memiliki setidaknya satu metode abstrak, yaitu metode yang dideklarasikan tetapi tidak diimplementasikan. Metode abstrak harus diimplementasikan oleh kelas turunan. Berbeda dengan kelas abstrak, semua metode yang berada pada interface harus merupakan abstract method.

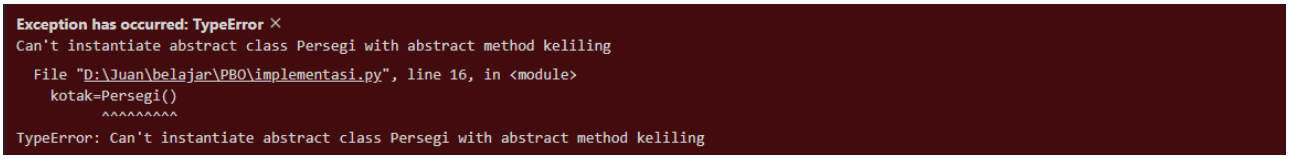
Interface dibagi menjadi 2, yaitu :

a. Informal Interface:
   Informal interface adalah sebuah konsep yang mendefinisikan sebuah kelas atau objek sebagai sebuah interface berdasarkan perilaku dan fungsi-fungsi yang diberikan.
b. Formal Interface:
   Konsep formal interface dapat diterapkan dalam Python dengan menggunakan modul abc (Abstract Base Class). Dalam Python, formal interface dapat didefinisikan sebagai kelas abstrak yang memiliki satu atau lebih metode abstrak. Metode abstrak adalah metode yang dideklarasikan tetapi tidak diimplementasikan dalam kelas abstrak tersebut. Kelas abstrak tidak dapat diinstansiasi dan hanya berfungsi sebagai kerangka kerja untuk kelas-kelas turunan yang mengimplementasi interface.

```
14. from abc import ABC, abstractmethod
15. 
16. class BangunDatar(ABC):
17.     @abstractmethod
18.     def luas(self):
19.         pass
20. 
21.     @abstractmethod
22.     def keliling(self):
23.         pass
24. 
```

Dapat dilihat telah dibuat sebuah abstract class bernama BangunDatar, class tersebut merupakan interface dikarenakan memiliki metode yang semuanya bersifat abstract. Dikarenakan itu semua keturunan dari class tersebut harus memiliki semua metode yang berada pada class BangunDatar jika tidak maka akan terjadi Error.

```
25. class Persegi(BangunDatar):
26.     def luas(self,panjang,lebar):
27.         return panjang*lebar
28. 
29. kotak=Persegi()
```



Dapat dilihat terjadi error pada program dikarenakan class Persegi belum memiliki semua method yang berada pada parentnya yaitu method keliling.

III. Metaclass

Metaclass di Python adalah sebuah kelas yang digunakan untuk membuat kelas baru. Dalam Python, setiap kelas sendiri juga merupakan sebuah objek, dan dibuat menggunakan sebuah metaclass bawaan yang dikenal sebagai ‘type’. Ketika kita membuat sebuah kelas baru, Python secara otomatis menggunakan ‘type’ sebagai metaclass untuk membuat kelas tersebut. Namun, kita dapat menentukan metaclass yang berbeda untuk membuat kelas baru dengan perilaku yang kita tentukan. Dalam hal ini, metaclass akan menentukan bagaimana kelas baru akan dibuat, termasuk cara mengatur atribut, metode, dan perilaku lainnya yang berkaitan dengan kelas tersebut. Metaclass memungkinkan kita untuk memodifikasi kelas Python secara dinamis saat dibuat. Kita dapat menambahkan atau menghapus atribut, metode, atau perilaku lainnya pada kelas yang dibuat menggunakan metaclass yang telah ditentukan. Dalam hal ini, metaclass bertindak sebagai "kelas kelas" yang mengontrol cara pembuatan kelas dan memungkinkan kita untuk menentukan perilaku khusus yang akan dimiliki oleh kelas tersebut.

```
30. buatkelas= type('Kelasbaru', (), {'dibuat':'kelas ini dibuat berdasarkan
    metaclass type' })
31.
32. print("Dibuat kelas baru yang memiliki nama", buatkelas)
33. print("Dari mana asal kelas ini?", buatkelas.dibuat)
```

Dibuat sebuah kelas baru menggunakan fungsi type, kelas tersebut memiliki nama Kelasbaru, dan memiliki atribut dibuat yang memiliki value ‘kelas ini dibuat berdasarkan metaclass type’

Sehingga output dari program tersebut adalah :

```
PS D:\Juan\belajar\PBO> & C:/Users/Jarvis/AppData/Local/Programs/Python/Python311/python.exe d:/Juan/belajar/PBO/implementasi.py
Dibuat kelas baru yang memiliki nama <class '__main__.Kelasbaru'>
Dari mana asal kelas ini? kelas ini dibuat berdasarkan metaclass type
PS D:\Juan\belajar\PBO>
```

IV. Kesimpulan :

1. Interface adalah kontrak antara dua bagian program yang memungkinkan mereka untuk berkomunikasi. Interface hanya mendefinisikan metode, tetapi tidak memberikan implementasi mereka. Oleh karena itu, kelas yang mengimplementasikan sebuah interface harus menyediakan implementasi untuk semua metode yang didefinisikan dalam interface. Kita perlu menggunakan interface ketika kita ingin membuat aplikasi yang modular dan scalable. Dengan menggunakan interface, kita dapat memastikan bahwa setiap class aplikasi dapat bekerja dengan baik tanpa harus mengetahui detail implementasi dari class lainnya. Interface juga memungkinkan kita untuk mengganti implementasi detail suatu class tanpa harus mengubah kode dari class lain yang terkait.

2. Kelas abstrak adalah sebuah kelas yang tidak bisa di-instansiasi secara langsung, melainkan digunakan sebagai kerangka dasar atau blueprint untuk kelas turunannya. Kelas abstrak ini memiliki minimal satu atau lebih metode abstrak yang harus di-implementasi pada kelas turunannya. Kita perlu menggunakan kelas abstrak ketika kita ingin membuat kerangka dasar untuk kelas-kelas turunannya, dan ingin memastikan bahwa setiap kelas turunan yang dihasilkan memiliki implementasi metode yang sama. Dengan menggunakan kelas abstrak, kita dapat memastikan bahwa setiap kelas turunan akan memiliki perilaku yang konsisten, karena mereka harus mengimplementasikan metode yang sama.

3. Perbedaan utama antara kelas abstrak dan interface adalah bahwa kelas abstrak dapat memiliki variabel anggota dan implementasi metode, sementara interface hanya mendefinisikan metode tanpa implementasi. Sehingga semua metode yang berada pada interface harus merupakan abstract method sedangakan pada kelas abstrak boleh memiliki metode biasa.

4. Kelas konkret (concrete class) adalah sebuah kelas yang dapat di-instansiasi secara langsung dan memiliki implementasi lengkap dari semua metode yang didefinisikan di dalamnya. Kelas konkret ini biasanya digunakan untuk membuat objek-objek yang sesuai dengan konsep yang didefinisikan oleh kelas tersebut. Kita perlu menggunakan kelas konkret ketika kita ingin membuat objek-objek konkret yang sesuai dengan konsep yang didefinisikan oleh kelas tersebut. Sebagai contoh, kita dapat membuat sebuah kelas konkret bernama "Persegi" yang memiliki atribut panjang dan lebar, dan metode untuk menghitung luas dan keliling. Dengan membuat objek konkret dari kelas "Persegi", kita dapat menghitung luas dan keliling dari sebuah persegi panjang dengan menggunakan atribut panjang dan lebar yang ada di dalam objek tersebut.

5. Dalam Python, metaclass adalah kelas yang digunakan untuk membuat kelas baru. Secara sederhana, metaclass adalah kelas dari kelas, yang digunakan untuk mengontrol perilaku kelas-kelas Python saat dibuat. Kita perlu menggunakan metaclass ketika kita ingin mengontrol bagaimana sebuah kelas dibuat dan berinteraksi dengan objek lainnya. Metaclass memungkinkan kita untuk menentukan atribut, metode, dan perilaku lainnya yang akan ditambahkan ke kelas saat dibuat. Dengan menggunakan metaclass, kita dapat menghasilkan kelas yang memiliki perilaku dan sifat yang berbeda dari kelas bawaan Python.

6. Perbedaan utama antara metaclass dan inheritance biasa adalah bahwa inheritance biasa menghasilkan kelas baru dengan mewarisi sifat-sifat dan perilaku dari kelas induknya, sedangkan metaclass menghasilkan kelas baru dengan mengontrol cara pembuatan kelas dan menambahkan perilaku khusus ke kelas baru tersebut. Metaclass memungkinkan kita untuk mengontrol seluruh proses pembuatan kelas, sementara inheritance hanya memungkinkan kita untuk mewarisi sifat dan perilaku yang telah ditentukan sebelumnya.

DAFTAR PUSTAKA :

https://www.petanikode.com/java-oop-abstract/

https://segalahal.com/2020/03/19/oop-java-abstract-class-part-6/

https://www.javatpoint.com/abstraction-in-python

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

https://www.pythontutorial.net/python-oop/python-metaclass/

Modul 6 Praktikum PBO ITERA