

**RESUME MATERI
PRAKTIKUM PBO RB
TUGAS MINGGU 6**

Oleh:

Attar Akram Abdillah (121140013)



Program Studi Teknik Informatika

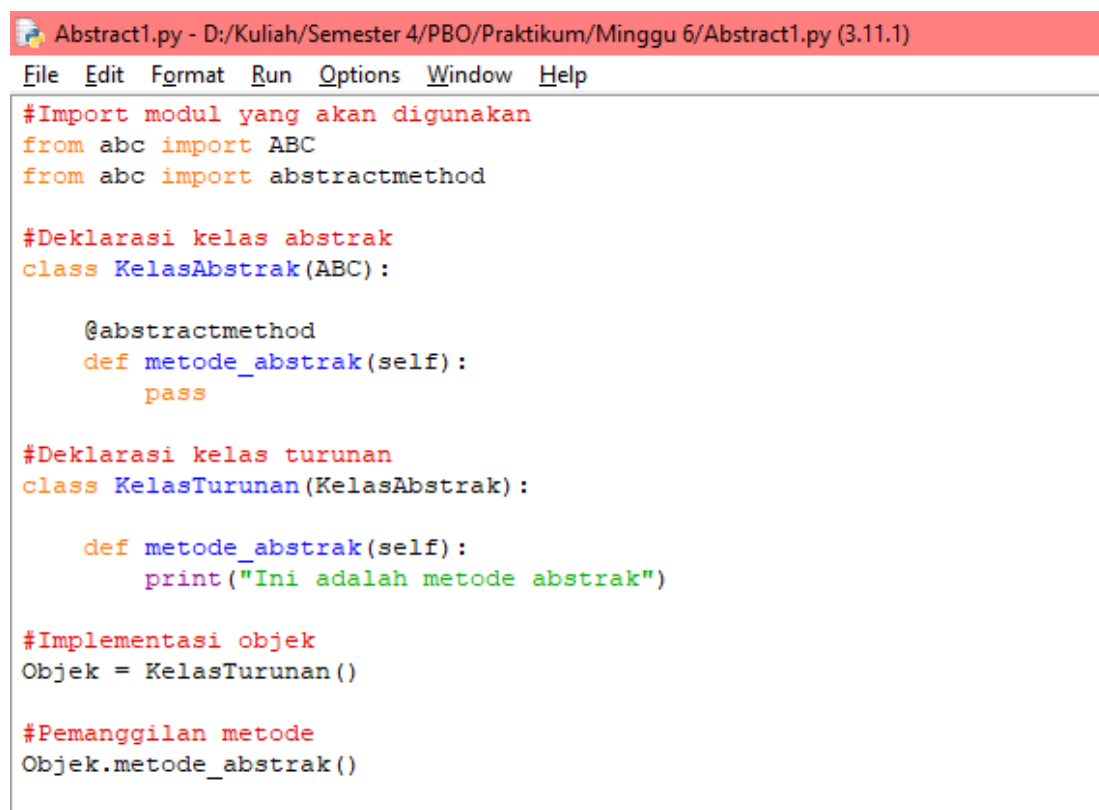
Institut Teknologi Sumatera

2023

1. Kelas Abstrak

Kelas abstrak adalah sebuah kelas yang menjadi landasan atau cetak biru dari kelas-kelas lain yang merupakan kelas turunan dari kelas abstrak tersebut [1]. Suatu kelas akan dikatakan sebagai kelas abstrak jika kelas tersebut merupakan kelas turunan dari kelas ABC dan memiliki setidaknya satu metode abstrak di dalamnya. Kelas abstrak ini memungkinkan untuk melakukan “pemaksaan” kepada kelas-kelas yang mewarisi kelas abstrak tersebut. Kelas abstrak akan memaksa kelas-kelas turunan untuk memiliki metode-metode abstrak yang telah dideklarasikan pada kelas abstrak.

Kelas abstrak sebenarnya tidak didukung oleh bahasa Python secara bawaan, oleh karena itu Python menghadirkan sebuah modul yang akan mendukung pengimplementasian kelas abstrak pada suatu kode. Modul yang dimaksud adalah modul *Abstract Base Classes* atau disingkat sebagai ABC [2]. Sehingga untuk membuat sebuah kelas abstrak kita cukup menambahkan modul ABC tersebut sebagai “orang tua” dari kelas abstrak yang akan dibuat. Dan juga untuk melakukan pendeklarasian metode abstrak pada suatu kelas abstrak, kita dapat menggunakan decorator “@abstractmethod” yang akan menandai sebuah metode sebagai metode abstrak.



```
Abstract1.py - D:/Kuliah/Semester 4/PBO/Praktikum/Minggu 6/Abstract1.py (3.11.1)
File Edit Format Run Options Window Help

#Import modul yang akan digunakan
from abc import ABC
from abc import abstractmethod

#Deklarasi kelas abstrak
class KelasAbstrak(ABC):

    @abstractmethod
    def metode_abstrak(self):
        pass

#Deklarasi kelas turunan
class KelasTurunan(KelasAbstrak):

    def metode_abstrak(self):
        print("Ini adalah metode abstrak")

#Implementasi objek
Objek = KelasTurunan()

#Pemanggilan metode
Objek.metode_abstrak()
```

Gambar 1.1. Contoh penerapan kelas abstrak pada python

Perlu diingat bahwasanya dalam penerapan dari kelas abstrak ini ada beberapa hal yang harus diperhatikan. Dalam pendeklarasian kelas turunan daripada kelas abstrak, tiap metode yang ada pada kelas abstrak haruslah dideklarasikan ulang pada kelas turunan.

```

#Deklarasi kelas abstrak
class KelasAbstrak(ABC):

    @abstractmethod
    def metode_abstrak(self):
        pass

#Deklarasi kelas turunan
class KelasTurunan(KelasAbstrak):

    def metode_bukan_abstrak(self):
        print("Ini adalah metode abstrak")

===== RESTART: D:/Kuliah/Semester 4/PBO/Praktikum/Minggu 6/Abstract1.py =====
Traceback (most recent call last):
  File "D:/Kuliah/Semester 4/PBO/Praktikum/Minggu 6/Abstract1.py", line 19, in <
module>
    Objek = KelasTurunan()
TypeError: Can't instantiate abstract class KelasTurunan with abstract method me
tode_abstrak

```

Gambar 1.2. Contoh error yang terjadi karena salahnya penerapan kelas abstrak

Dan juga perlu diingat bahwa kita tidak dapat melakukan implementasi objek terhadap kelas abstrak [3]. Namun meski demikian kita masih bisa menambahkan konstruktor kedalam suatu kelas abstrak yang kita deklarasikan.

```

Abstract1.py - D:/Kuliah/Semester 4/PBO/Praktikum/Minggu 6/Abstract1.py (3.11.1)
File Edit Format Run Options Window Help

#Import modul yang akan digunakan
from abc import ABC
from abc import abstractmethod

#Deklarasi kelas abstrak
class KelasAbstrak(ABC):
    def __init__(self, atribut):
        self.atribut = atribut

    @abstractmethod
    def metode_abstrak(self):
        pass

#Deklarasi kelas turunan
class KelasTurunan(KelasAbstrak):

    def metode_abstrak(self):
        print("Ini adalah metode abstrak dengan atribut " + self.atribut)

#Implementasi objek
Objek = KelasTurunan("A")

#Pemanggilan metode
Objek.metode_abstrak()

#Implementasi objek ke kelas abstrak (error)
Objek2 = KelasAbstrak("A")

```

Gambar 1.3. Contoh implementasi objek ke kelas abstrak

```

> ===== RESTART: D:/Kuliah/Semester 4/PBO/Praktikum/Minggu 6/Abstract1.py =====
> Ini adalah metode abstrak dengan atribut A
> Traceback (most recent call last):
>   File "D:/Kuliah/Semester 4/PBO/Praktikum/Minggu 6/Abstract1.py", line 27, in <
> module>
>     Objek2 = KelasAbstrak("A")
> TypeError: Can't instantiate abstract class KelasAbstrak with abstract method me
> tode_abstrak
>

```

Gambar 1.4. Hasil keluaran kode pada gambar 1.3.

2. Interface

Interface adalah suatu metode yang digunakan untuk menetapkan “kontrak” dalam pembuatan kelas yang ada pada suatu kode [4]. Interface pada Python sebenarnya pada dasarnya sama saja dengan kelas abstrak yang telah dijelaskan sebelumnya, hanya saja faktor kunci yang membedakan dari kelas abstrak biasa interface hanya dapat memiliki metode abstrak di dalamnya.

```

Interface.py - D:/Kuliah/Semester 4/PBO/Praktikum/Minggu 6/Interface.py (3.11.1)
File Edit Format Run Options Window Help

#Import modul yang akan digunakan
from abc import ABC
from abc import abstractmethod

#Deklarasi interface
class Interface(ABC):
    @abstractmethod
    def kontrak1(self):
        pass

    @abstractmethod
    def kontrak2(self):
        pass

#Deklarasi kelas turunan 1
class Turunan1(Interface):
    def kontrak1(self):
        print("Kontrak 1 Turunan 1")

    def kontrak2(self):
        print("Kontrak 2 Turunan 1")

#Deklarasi kelas turunan 2
class Turunan2(Interface):
    def kontrak1(self):
        print("Kontrak 1 Turunan 2")

    def kontrak2(self):
        print("Kontrak 2 Turunan 2")

#Implementasi objek
Objek1 = Turunan1()
Objek2 = Turunan2()

#Pemanggilan metode
Objek1.kontrak1()
Objek1.kontrak2()
Objek2.kontrak1()
Objek2.kontrak2()

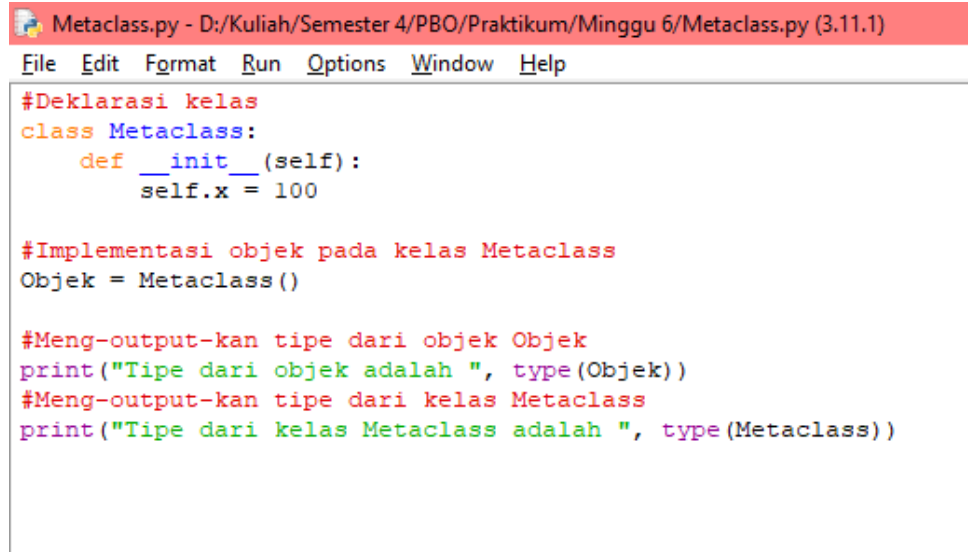
```

Gambar 1.5. Contoh penerapan konsep interface pada Python

Selain dari contoh yang telah diberikan, penerapan dari interface pada Python bisa juga dilakukan secara informal, yaitu dengan menggunakan metaclass atau virtual base class.

3. Metaclass

Metaclass adalah sebuah kelas daripada kelas lainnya yang mendefinisikan bagaimana kelas-kelas lainnya akan bekerja [5]. Adanya metaclass ini dikarenakan Python tidak memiliki tipe dasar sebagaimana bahasa-bahasa pemrograman lainnya, Python hanya memiliki objek atau kelas didalamnya. Oleh karena itu, Python disertai metaclass yang merupakan suatu kelas bawaan yang berperan sebagai pengganti dari tipe dasar pada bahasa lain.

A screenshot of a Python IDE window titled 'Metaclass.py - D:/Kuliah/Semester 4/PBO/Praktikum/Minggu 6/Metaclass.py (3.11.1)'. The window contains the following Python code:

```
#Deklarasi kelas
class Metaclass:
    def __init__(self):
        self.x = 100

#Implementasi objek pada kelas Metaclass
Objek = Metaclass()

#Meng-output-kan tipe dari objek Objek
print("Tipe dari objek adalah ", type(Objek))
#Meng-output-kan tipe dari kelas Metaclass
print("Tipe dari kelas Metaclass adalah ", type(Metaclass))
```

Gambar 1.6. Contoh implementasi metaclass

4. Kesimpulan

Kelas abstrak merupakan sebuah cetak biru yang menjadi landasan dibangunnya kelas-kelas turunannya. Interface pula kurang lebih sama dengan kelas abstrak dimana keduanya sama-sama berperan sebagai cetak biru dari kelas turunannya. Perbedaan diantara keduanya adalah interface hanya dapat memiliki metode abstrak didalamnya, karena memang fungsi dari interface juga yang menjadi suatu “kontrak”. Kelas abstrak dapat digunakan ketika adanya fitur umum yang dimiliki semua objek dan interface dapat digunakan ketika tiap fitur harus diimplementasikan secara berbeda pada tiap objek. Dikenal pula kelas konkret yaitu kelas yang tidak memiliki satu pun metode abstrak. Kelas konkret digunakan ketika kita ingin mengimplementasikan suatu objek ke kelas tersebut.

Metaclass adalah kelas bawaan dari Python yang berperan sebagai tipe dasar (int, float, char, dst) suatu variabel, objek, atau kelas. Metaclass ini digunakan ketika diperlukannya penetapan tipe dasar suatu kelas. Metaclass berbeda dengan inheritance biasa dikarenakan metaclass tidak akan menjadi bagian dari hirarki kelas objek, dibanding dengan kelas biasa yang akan tergabung kedalam hirarki tersebut.

Daftar Pustaka

- [1] bestharadhakrishna, "Abstract Classes in Python," GeeksforGeeks, 19 March 2021. [Online].
] Available: <https://www.geeksforgeeks.org/abstract-classes-in-python/>. [Accessed 11 April 2023].
- [2] G. John, "Abstract Base Classes in Python (abc)," tutorialspoint, 30 July 2019. [Online]. Available:
] <https://www.tutorialspoint.com/abstract-base-classes-in-python-abc>. [Accessed 11 April 2023].
- [3] A. N., "How to Use Abstract Classes in Python," towardsdatascience.com, 17 October 2021.
] [Online]. Available: <https://towardsdatascience.com/how-to-use-abstract-classes-in-python-d4d2ddc02e90#:~:text=An%20abstract%20class%20is%20a,they%20are%20expected%20to%20have..> [Accessed 12 April 2023].
- [4] W. Murphy, "Implementing an Interface in Python," Real Phyton, [Online]. Available:
] <https://realpython.com/python-interface/>. [Accessed 12 April 2023].
- [5] D. Mwiti, "Introduction to Python Metaclasses," Datacamp, December 2018. [Online]. Available:
] <https://www.datacamp.com/tutorial/python-metaclasses>. [Accessed 12 April 2023].