

**TUGAS 6 PRAKTIKUM
PEMORGRAMAN BERORIENTASI OBJEK**



Disusun oleh:
Andreyan Renaldi (121140186)

**JURUSAN TEKNOLOGI PRODUKSI DAN INDSUTRI
PROGRAM STUDI TEKNIK INFORMATIKA
INSTITUT TEKNOLOGI SUMATERA
2023**

DAFTAR ISI

A. Abstract

Abstraction adalah pilar terakhir dari ke-empat pilar pemrograman berorientasi objek yang di bahas pada artikel ini. Abstraction memiliki tujuan untuk menyembunyikan detail yang tidak terlalu penting dari user. Kita dapat membayangkannya seperti sebuah smartphone, dimana untuk menggunakan device tersebut, kita tidak perlu memahami apa yang terjadi di dalam logika elektronik dan software di dalamnya. Namun, kita hanya perlu menekan tombol power dan menyentuh layar dengan menggunakan tangan kita agar alat tersebut bisa bekerja dengan semestinya.

Python memiliki modul untuk menggunakan Abstract Base Classes (ABC). Modulnya bernama abc. Contoh kode di bawah ini adalah cara penggunaan modul abc untuk mendefinisikan abstraction.

Untuk menggunakan metode abstrak, kita harus menambahkan dekorator pada metode yang akan dijadikan abstrak dengan cara menambahkan `@abstractmethod` di atas metode tersebut. Untuk menunjukkan bahwa metode tersebut adalah abstrak. Mari kita coba untuk menjalankan code ini.

```
D: > Python > 📄 coba-coba2.py > ...
1  from abc import ABC, abstractmethod
2
3  class BangunDatar(ABC):
4      @abstractmethod
5      def luas(self):
6          return self.__sisi * self.__sisi
7
8      @abstractmethod
9      def keliling(self):
10         return 4 * self.__sisi
11
12     class Persegi(BangunDatar):
13         def __init__(self, sisi):
14             self.__sisi = sisi
15         def luas(self):
16             return self.__sisi * self.__sisi
17         def keliling(self):
18             return 4 * self.__sisi
19
20
21     persegi = Persegi(4)
22     print(persegi.luas())
23     print(persegi.keliling())
```

Output-nya yaitu:

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL

PS D:\Python\TUBES PBO> & C:/Users/ACER/AppData/Local/Microsoft
Python/coba-coba2.py
16
16
PS D:\Python\TUBES PBO>
```

B. Interface

Interface adalah koleksi dari method atau fungsi-fungsi yang perlu disediakan oleh implementing class (child class). Interface mengandung metode yang bersifat abstract. Metode abstract akan memiliki satu-satunya deklarasi karena tidak adanya implementasi. Pengimplementasian sebuah interface adalah sebuah cara untuk menulis kode yang elegan dan terorganisir.

```
D: > Python > 📄 coba-coba2.py > ...
1  class Prodi:
2      def __init__(self, prodi):
3          self.__daftar_prodi = prodi
4
5      def __len__(self):
6          return len(self.__daftar_prodi)
7
8      def __contains__(self, prodi):
9          return prodi in self.__daftar_prodi
10
11 class ptn(Prodi):
12     def __init__(self, prodi):
13         super().__init__(prodi)
14
15     def __iter__(self):
16         return iter(self._Prodi__daftar_prodi)
17
18 itera = ptn(["informatika", "elektro", "sipil"])
19 print(len(itera))
20 print("informatika" in itera)
21 print("sipil" not in itera)
22 print("kedokteran" in itera)
23 print()
24 for prodi in itera:
25     print(prodi)
```

```
PROBLEMS    OUTPUT    DEBUG CONSOLE    TERMINAL

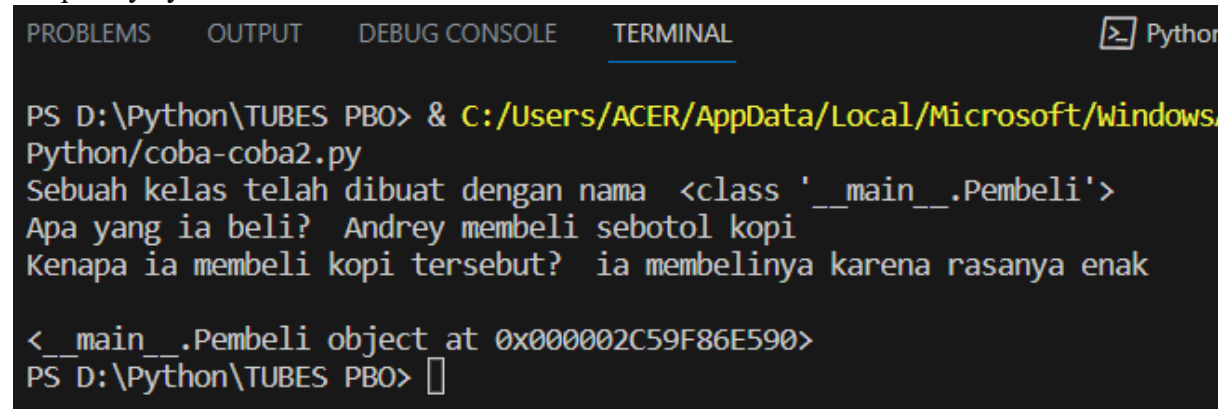
Python/coba-coba2.py
3
True
False
False

informatika
elektro
sipil
PS D:\Python\TUBES PBO>
```

Metaclass adalah konsep OOP, bersembunyi di balik hampir semua kode Python. Apakah Anda menyadarinya atau tidak. Namun, ketika diperlukan, Python menyediakan kemampuan yang tidak didukung oleh semua bahasa berorientasi objek: Anda dapat membukanya dan menentukan metaclass khusus.

```
D: > Python > coba-coba2.py > ...
1  nama_class = type('Pembeli',
2  |           |   (), {'beli': 'Andrey membeli sebotol kopi',
3  |           |   |   'alasan': 'ia membelinya karena rasanya enak'})
4
5  print("Sebuah kelas telah dibuat dengan nama ", nama_class)
6  print("Apa yang ia beli? ", nama_class.beli)
7  print("Kenapa ia membeli kopi tersebut? ", nama_class.alasan)
8  print()
9  print(nama_class())
```

Output-nya yaitu:



```
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL Python
PS D:\Python\TUBES PBO> & C:/Users/ACER/AppData/Local/Microsoft/Windows
Python/coba-coba2.py
Sebuah kelas telah dibuat dengan nama <class '__main__.Pembeli'>
Apa yang ia beli? Andrey membeli sebotol kopi
Kenapa ia membeli kopi tersebut? ia membelinya karena rasanya enak

<__main__.Pembeli object at 0x000002C59F86E590>
PS D:\Python\TUBES PBO> 
```

D. Kesimpulan

Interface merupakan sekumpulan metode atau fungsi yang harus ada dalam suatu kelas, tetapi tidak memberikan implementasi konkret untuk metode tersebut. Interface mengandung metode yang bersifat abstract. Metode abstract akan memiliki satu-satunya deklarasi karena tidak adanya implementasi. Pengimplementasian sebuah interface adalah sebuah cara untuk menulis kode yang elegan dan terorganisir. Interface digunakan ketika kita ingin menggambarkan perilaku yang harus dimiliki oleh suatu objek atau kelas.

Abstraction adalah pilar terakhir dari ke-empat pilar pemrograman berorientasi objek yang di bahas pada artikel ini. Abstraction memiliki tujuan untuk menyembunyikan detail yang tidak terlalu penting dari user. Kita dapat membayangkannya seperti sebuah smartphone, dimana untuk menggunakan device tersebut, kita tidak perlu memahami apa yang terjadi di dalam logika elektronik dan software di dalamnya. Namun, kita hanya perlu menekan tombol power dan menyentuh layar dengan menggunakan tangan kita agar alat tersebut bisa bekerja dengan semestinya. Abstract class digunakan ketika kita ingin membuat kelas yang tidak dapat diinstansiasi secara langsung, tetapi harus diwarisi oleh kelas-kelas turunannya.

Perbedaan antara interface dan abstract class yaitu sbb.

1. Semua metode dari sebuah formal interface adalah abstract, sedangkan pada abstract class dapat mempunyai metode abstract ataupun metode konkrit.
2. Interface digunakan jika semua fitur perlu untuk diimplementasikan secara berbeda untuk objek yang berbeda, sedangkan abstract class digunakan Ketika ada beberapa fitur umum yang dimiliki oleh semua objek.
3. Interface cenderung lebih lambat dibandingkan dengan abstract class.

Kelas konkret adalah kelas yang memiliki implementasi lengkap untuk semua metode yang didefinisikan dalam kelas tersebut. Dalam arti lain, kelas konkret tidak memiliki metode abstrak yang harus diimplementasikan oleh kelas turunan. Sebaliknya, kelas konkret menyediakan implementasi konkret untuk semua metode yang didefinisikan. Kita perlu menggunakan kelas konkret ketika kita ingin membuat objek yang dapat diinstansiasi secara langsung dan memiliki implementasi lengkap untuk semua metode yang diperlukan. Kelas konkret biasanya digunakan untuk menggambarkan objek nyata atau entitas dalam sistem yang sedang kita bangun.

Metaclass adalah konsep OOP, bersembunyi di balik hampir semua kode Python. Apakah Anda menyadarinya atau tidak. Namun, ketika diperlukan, Python menyediakan kemampuan yang tidak didukung oleh semua bahasa berorientasi objek. Metaclass dalam Python adalah sebuah kelas yang digunakan untuk membuat dan mengontrol perilaku kelas lainnya. Metaclass memungkinkan pengguna untuk memanipulasi pembuatan kelas, seperti mengubah perilaku warisan, mengubah atribut kelas, dan mengubah metode yang diturunkan. Metaclass dapat kita gunakan ketika kita ingin mempengaruhi perilaku pembuatan kelas, pengubahan atribut atau metode, atau melaksanakan tugas khusus lainnya yang berkaitan dengan kelas.

Perbedaan antara metaclass dan inheritance, yaitu Inheritance (pewarisan) adalah mekanisme yang digunakan untuk mewarisi perilaku dan atribut dari satu kelas ke kelas lain dalam hierarki pewarisan. Dalam inheritance, kelas turunan mendapatkan metode dan atribut dari kelas induknya. Sementara itu, metaclass adalah kelas yang digunakan untuk mengendalikan pembuatan dan perilaku kelas-kelas. Metaclass mempengaruhi pembuatan kelas dan dapat memodifikasi atau menambahkan perilaku kelas secara otomatis sebelum objek kelas dibuat