LAPORAN
KELAS ABSTRAK, INTERFACE DAN METACLASS

Disusun Oleh :
Ryan Ernanda
120140154
Praktikum PBO RB

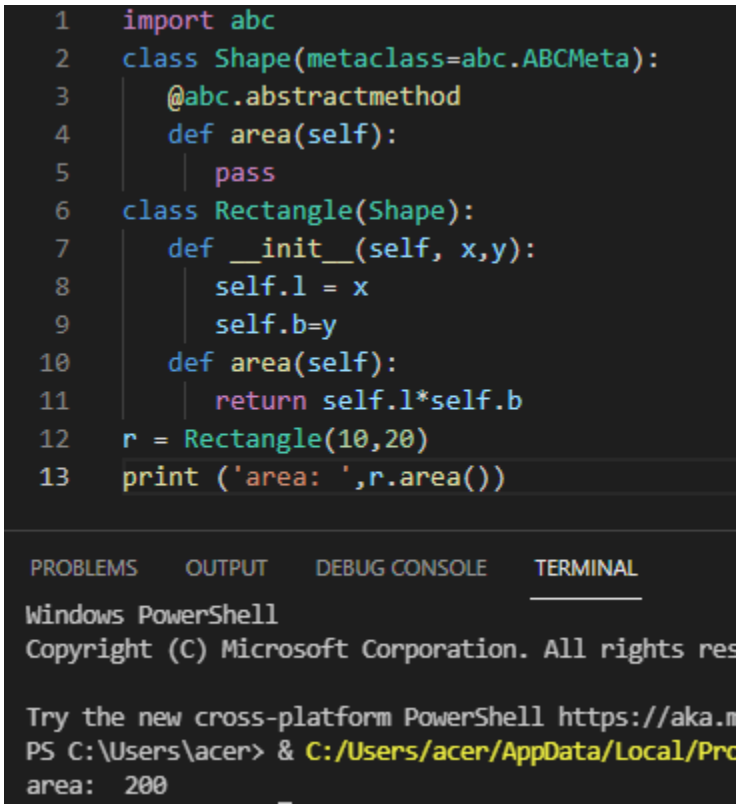
INSTITUT TEKNOLOGI SUMATERA
JURUSAN TEKNOLOGI PRODUKSI DAN INDUSTRI
LAMPUNG SELATAN
2022

RINGKASAN

A. Kelas Abstrak

Kelas abstrak adalah kelas yang tidak dapat dipakai (tidak dapat diubah menjadi objek) dan bertindak sebagai "bingkai dasar" untuk kelas turunannya. Dalam kelas abstrak biasanya akan ada metode abstrak. Metode abstrak adalah "metode dasar" yang harus diimplementasikan kembali dalam subkelas. Kelas abstrak digunakan dalam pewarisan untuk "memaksa" penerapan metode yang sama untuk semua kelas yang diturunkan dari kelas abstrak.

Kelas abstrak digunakan untuk menyusun logika turunan dalam pemrograman objek. Ada beberapa konsep pemrograman kelas abstrak yang tidak dapat diadaptasi sehingga ketika merancang program abstrak, diperlukan cara khusus untuk menyelesaikannya, terutama karena inisialisasi adalah salah satu proses terpenting. Di luar langkah inisialisasi dalam mendesain objek pada abstrak, menjadi tidak mungkin karena objek yang ingin dikonsep juga abstrak. Untuk mengatasi masalah ini perlu menggunakan konsep pewarisan (inheritance), dengan konsep pewarisan kelas abstrak yang membantu sebagai superclass yang dapat dipanggil pada fungsi yang ingin digunakan. Selain konsep pada bagian sebelumnya, kelas abstrak ditandai dengan pewarisan dari kelas ABC (abstract base class) dan pada bagian fungsi ditandai dengan @abstractmethod pada setiap fungsi abstrak, beserta contoh penerapan konsep abstrak. @abstractmethod adalah dekorator yang mengubah metode normal menjadi metode abstrak. Berikut merupakan contoh dari program kelas abstrak :



```
1   import abc
2   class Shape(metaclass=abc.ABCMeta):
3       @abc.abstractmethod
4       def area(self):
5           pass
6   class Rectangle(Shape):
7       def __init__(self, x,y):
8           self.l = x
9           self.b=y
10      def area(self):
11          return self.l*self.b
12  r = Rectangle(10,20)
13  print ('area: ',r.area())
```

```
PROBLEMS   OUTPUT   DEBUG CONSOLE   TERMINAL
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights res

Try the new cross-platform PowerShell https://aka.m
PS C:\Users\acer> & C:/Users/acer/AppData/Local/Prc
area:  200
```

Modul abc mendefinisikan kelas ABCMeta sebagai metaclass untuk mendefinisikan kelas dasar abstrak. Kelas form memiliki metode area() yang didekorasi oleh abstractmethod.Kelas Rectangle sekarang menggunakan kelas Shape di atas sebagai superclass-nya dan mengimplementasikan metode abstract area().

B. Interface

Interface (antarmuka) merupakan *blue print* dari class. Isi dari method-nya ini kosong dan akan diimplementasikan pada class lain. *interface* bertindak sebagai penghubung antara sesuatu yang 'abstrak' dan sesuatu yang nyata. Konsep interface pemrograman berorientasi objek (Object Orientet Programming), digunakan sebagai model untuk perancangan kelas. *interface* juga mendefinisikan metode dan didasarkan pada konsep yang sama dengan kelas, sedangkan kelas dan *interface* berisi metode non-abstrak. Metode abstrak yang dapat digunakan dalam membangun *interface* adalah metode yang tidak memiliki implementasi atau tidak memiliki badan. Jadi *interface* hanya mendefinisikan metode abstrak tanpa implementasi. Implementasi metode abstrak ini ditentukan oleh kelas yang mengimplementasikan *interface*. *interface* dapat digunakan untuk menentukan kontrak Dalam bahasa pemrograman python, konsep *interface* terdiri dari konsep pewarisan berganda, hal ini terjadi karena python tidak memiliki fitur khusus dalam mengimplementasikan konsep *interface* dan implementasi *interface*. Berikut merupakan contoh dari program informal interface :

```
1   class Fruits :
2       def __init__( self, ele):
3           self.__ele = ele
4       def __contains__( self, ele):
5           return ele in self.__ele
6       def __len__( self ):
7           return len( self.__ele)
8   Fruits_list = Fruits([ "Apple", "Banana", "Orange" ])
9   # protocol to get size
10  print(len(Fruits_list))
11  # protocol to get container
12  print("Apple" in Fruits_list)
13  print("Mango" in Fruits_list)
14  print("Orange" not in Fruits_list)
```

```
PROBLEMS    OUTPUT    DEBUG CONSOLE    TERMINAL

Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

Try the new cross-platform PowerShell https://aka.ms/pscore6

PS C:\Users\acer> & C:/Users/acer/AppData/Local/Programs/Python/Pyt
3
True
False
False
```

Seperti pada kode contoh di atas, class Fruits mengimplementasikan metode __len__, dan __contains__, jadi pada instance class Fruits, kita bisa langsung menggunakan fungsi len untuk mendapatkan ukuran dan bisa mengecek keanggotaan dengan menggunakan operator in. Seperti pada kode di atas, metode __iter__ (protokol iterable) tidak diterapkan, jadi tidak akan mengulangi instance Buah. Oleh karena itu *interface* informal tidak dapat ditegakkan secara formal.

C. Metaclass

Metaclass adalah kelas yang instance-nya adalah kelas-kelas lain. Sama seperti kelas biasa yang perilakunya sama seperti instance kelas. Metaclass merupakan kelas yang mewarisi dari "tipe". Perbedaan lainnya adalah meteclass dipanggil secara otomatis ketika pernyataan kelas menggunakan metaclass berakhir, yang berarti jika tidak ada kata kunci "metaclass" yang dilewatkan setelah kelas dasar (mungkin tanpa kelas dasar) dari header kelas, type() (yaitu ketik __call__) akan dipanggil. Namun, jika kata kunci metaclass digunakan, kelas yang ditugaskan padanya akan dipanggil alih-alih tipenya. Metaclass di Python mendefinisikan perilaku instance kelas. Metode yang harus diterapkan metaclass adalah __new__method. Argumen yang disebutkan dalam __new__method adalah:

1. Argumen pertama adalah metaclass itu sendiri
2. Argumen kedua adalah nama kelas
3. Argumen ketiga adalah superclass (sebagai tuple)
4. Argumen keempat adalah atribut class (sebagai kamus)

Berikut merupakan contoh dari program Metaclass:

```python
# metaclass
class MultiBases(type):
    # overriding __new__ method
    def __new__(cls, clsname, bases, clsdict):
        # jika tidak ada kelas dasar yang lebih besar dari 1
        # raise error
        if len(bases)>1:
            raise TypeError("Mewarisi beberapa kelas dasar!")

        # metode __new__ dari kelas super.
        # Panggil __init__ dari tipe kelas
        return super().__new__(cls, clsname, bases, clsdict)

# metaclass dapat ditentukan dengan argumen kata kunci 'metaclass'
# sekarang kelas MultiBase digunakan untuk membuat kelas
# base ini disebar ke semua subclass dari Base
class Base(metaclass=MultiBases):
    pass
# bukan raise error
class A(Base):
    pass
# bukan raise error
class B(Base):
    pass
# ini raise error!
class C(A, B):
    pass

```

```
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

Try the new cross-platform PowerShell https://aka.ms/pscore6

PS C:\Users\acer> & C:/Users/acer/AppData/Local/Programs/Python/Python310/python.exe "d:/1. KULIAH/2.tugas kuliah/Semester 4/pbo/Metaclass.py"
Traceback (most recent call last):
  File "d:\1. KULIAH\2.tugas kuliah\Semester 4\pbo\Metaclass.py", line 26, in <module>
    class C(A, B):
  File "d:\1. KULIAH\2.tugas kuliah\Semester 4\pbo\Metaclass.py", line 8, in __new__
    raise TypeError("Mewarisi beberapa kelas dasar!")
TypeError: Mewarisi beberapa kelas dasar!
PS C:\Users\acer>
```

Ada beberapa masalah yang dapat diselesaikan oleh dekorator (dengan mudah) dan juga oleh metaclass. Tetapi ada beberapa masalah yang hasilnya hanya dapat dicapai oleh metaclass. untuk men-debug metode kelas adalah setiap kali metode kelas dijalankan, ia harus mencetak nama yang sepenuhnya memenuhi syarat sebelum menjalankan tubuhnya.

KESIMPULAN

- *Interface* (antarmuka) merupakan *blue print* dari class. Isi dari method-nya ini kosong dan akan diimplementasikan pada class lain. *interface* bertindak sebagai penghubung antara sesuatu yang ‘abstrak’ dan sesuatu yang nyata. Interface ini digunakan saat kita perlu membuat model untuk perancangan kelas.
- Kelas abstrak adalah kelas yang tidak dapat dipakai (tidak dapat diubah menjadi objek) dan bertindak sebagai "bingkai dasar" untuk kelas turunannya. Kelas abstrak digunakan ketika ingin menyusun logika perwarisan. Perbedaan dengan interface yaitu interface hanya dapat berisi abstrak method, dalam kecepatan eksekusinya interface relatif lebih lambat, abstrak memiliki access specifier berbeda dengan interface yang tidak memilikinya karena interface bersifat public, dan interface hanya berisi signature.
- Konkret biasanya dipakai dalam membuat instance objek dan biasa disebut tubuhnya konstruktor, tetapi decorator yang biasa digunakan juga merupakan konkret. Konkret ini digunakan ketika kita membuat konstruktor dan decorator didalam fungsi.
- Metaclass adalah kelas yang instance-nya adalah kelas-kelas lain. Metaclass merupakan kelas yang mewarisi dari "type". Metaclass biasa digunakan sebagai pabrik kelas, dan dalam pembuatan objek dengan memanggil kelas, python membuat kelas baru dengan memanggil metaclass tersebut. Bedanya metaclass dan inheritance yaitu inheritance merupakan perwarisan dan penggunaanya sangat dibutuhkan ketika ada kode yang sama dan perlu ditulis berulang ulang, sedangkan metaclass merupakan barang yang menciptakan kelas atau biasanya disebut dengan pabrik kelas

DAFTAR PUSTAKA

[1] Andre, "duniailkom.com," 10 Oktober 2014. [Online]. Available: https://www.duniailkom.com/tutorial-belajar-oop-php-pengertian-abstract-class-dan-abstract-method-php/.

[2] A. Muhardin, "petanikode.com," 28 Desember 2019. [Online]. Available: https://www.petanikode.com/java-oop-interface/.

[3] B. Klein, "Metaclasses," 1 Februari 2022. [Online]. Available: https://python-course.eu/oop/metaclasses.php.

[4] ITERA, Tim Pengajar PBO, Kelas Abstrak dan Interface, Lampung Selatan, 2022.

[5] W. Murphy, "Implementing an Interface in Python," realpython.com, 2020. [Online]. Available: https://realpython.com/python-interface/.

[6] G. John, "Abstract Base Classes in Python (abc)," tutorialspoint.com, 17 January 2019. [Online]. Available: https://www.tutorialspoint.com/abstract-base-classes-in-python-abc.

[7] Andre, "Tutorial Belajar OOP PHP Part 9: Pengertian Inheritance (Pewarisan)," duniailkom.com, 3 Oktober 2014. [Online]. Available: https://www.duniailkom.com/tutorial-belajar-oop-php-pengertian-inheritance-pewarisan/.