PRAKTIKUM PEMROGRAMAN BERBASIS OBJEK

TUGAS 6

Disusun oleh:

Inori Muira Sitanggang

121140020

RB

PROGRAM STUDI TEKNIK INFORMATIKA

INSTITUT TEKNOLOGI SUMATERA

LAMPUNG SELATAN 2023

A. Abstraksi dalam Python

Konsep abstraksi ini memungkinkan kita untuk menyembunyikan detail yang tidak relevan dan fokus pada informasi penting yang relevan. Dalam Python, abstraksi dapat diterapkan melalui kelas. Kelas Abstrak adalah sebuah class yang tidak bisa diinstansiasi (tidak bisa dibuat menjadi objek) langsung dan berperan sebagai 'kerangka dasar' bagi class turunannya. Di dalam kelas abstrak umumnya akan memiliki sebuah abstract method. Abstract method adalah sebuah method yang harus diimplementasikan ulang di dalam class anak (child class). Abstract method ditulis tanpa isi dari method, melainkan hanya 'signature'-nya saja. Signature dari sebuah method adalah bagian method yang terdiri dari nama method dan parameternya (jika ada).

- Implementasi Kelas Abstrak dengan Modul ABC

  Python memiliki modul untuk menggunakan Abstract Base Classes (ABC). Fungsi abstrak pada kelas abstrak ditandai dengan memberikan decorato @abstractmethod pada fungsi yang dibuat.

```
from abc import ABC, abstractmethod

class Keluarga(ABC):
    @abstractmethod
    def bahagia(self):
        pass

    @abstractmethod
    def brokenhome(self):
        pass

Jono=Keluarga()
```

Terjadi *error* ketika kelas tersebut langsung dijadikan sebuah objek. Untuk mengatasi hal tersebut dan melakukan implementasi terhadap kelas abstrak diperlukan sebuah kelas 'child' dari kelas abstrak tersebut.

```
from abc import ABC, abstractmethod

class Keluarga(ABC):
    @abstractmethod
    def bahagia(self):
        pass

    @abstractmethod
    def data(self):
        pass

class Jono(Keluarga):
    def __init__(self, asal, umur):
        self.asal=asal
        self.umur=umur

    def bahagia(self):
        print("Asal: ", self.asal)
        print("Umur: ", self.umur)

    def data(self):
        print(f"{self.asal} adalah tempat kelahiran")

Man=Jono("Tanggamus", "45")
Man.bahagia()
Man.data()
```

```
input
Asal:  Tanggamus
Umur:  45
Tanggamus adalah tempat kelahiran
```

Ketika membuat kelas yang tidak menggunakan modul ABC, kelas abstrak ini dapat menggunakan konstruktor yang isi metodenya tidak selalu bernilai *pass*, karena dapat menggunakan fungsi biasa.

```python
from abc import ABC, abstractmethod

class Keluarga(ABC):
    def __init__(self, asal, umur):
        self.asal=asal
        self.umur=umur

    @abstractmethod
    def bahagia(self):
        pass

    @abstractmethod
    def data(self):
        pass

class Jono(Keluarga):
    def __init__(self, asal, umur):
        super().__init__(asal,umur)

    def bahagia(self):
        print("Asal: ", self.asal)
        print("Umur: ", self.umur)

    def data(self):
        print(f"{self.asal} adalah tempat kelahiran")

Man=Jono("Tanggamus", "45")
Man.bahagia()
Man.data()
```

input

```
Asal:  Tanggamus
Umur:  45
Tanggamus adalah tempat kelahiran
```

B. Interface

Interface dalam python adalah koleksi dari method atau fungsi-fungsi yang perlu deisediakan oleh implementing class atau *child class*. Sleian itu, interface ini mengandung metode yang bersifat abstract, serta pengimplementasiannya. Terdapat 2 Interface, yaitu:

1. Interface Formal
   Pada formal interface
   kelas parent (abstrak) dapat dibangun hanya dengan sedikit baris kode, untuk kemudian diimplementasikan pada kelas turunannya (konkret). Implementasi ini bersifat wajib/dipaksakan sehingga dinamakan formal interface.

```python
from abc import ABC, abstractmethod

class Keluarga(ABC):
    @abstractmethod
    def bahagia(self):
        pass

    @abstractmethod
    def data(self):
        pass

class Jono(Keluarga):
    def __init__(self, asal, umur):
        self.asal=asal
        self.umur=umur

    def bahagia(self):
        print("Asal: ", self.asal)
        print("Umur: ", self.umur)

    def data(self):
        print(f"{self.asal} adalah tempat kelahiran")

Man=Jono("Tanggamus", "45")
Man.bahagia()
Man.data()
```

input

```
Asal:  Tanggamus
Umur:  45
Tanggamus adalah tempat kelahiran
```

2. Interface Informal
   Informal interface adalah kelas yang mendefinisikan metode atau fungsi-fungsi yang bisa di override/implementasi namun tanpa adanya unsur paksaan (cukup diimplementasi hanya bila dibutuhkan). Untuk mengimplementasikannya kita harus membuat kelas konkrit.

Kelas konkrit adalah subclass dari (abstrak) interface yang menyediakan implementasi dari method atau fungsi-fungsi di kelas interface.

```
from abc import ABC, abstractmethod

class Keluarga(ABC):
    def __init__(self, manusia):
        self.__manusia=manusia

    def __len__(self):
        return len(self.__manusia)

    def __contains__(self, manusia):
        return manusia in self.__manusia

class Jumlah(Manusia):
    pass

Keluarga_jono = Jumlah(["Ayah", "Ibu", "Anak"])
print(len(Keluarga_jono))
print("Ayah" in Keluarga_jono)
print("Dodol" in Keluarga_jono)
```

C. Metaclass

Pada Python semua tipe pada dasarnya adalah suatu objek/kelas juga (termasuk int, float, dll), tidak seperti bahasa pemrograman yang lain. Metaclass adalah konsep OOP yang ada dalam semua kode Python secara default. Setiap class yang dibuat di Python adalah sebuah instance dari tipe metaclass. Fungsi type() dapat membuat class secara dinamis karena memanggil type() akan membuat instance baru dari metaclass type. Contoh implementasi menggunakan metaclass type:

```
nama="Jono"
umur=45
tinggi=172.45

print(type(nama))
print(type(umur))
print(type(tinggi))
```

```
<class 'str'>
<class 'int'>
<class 'float'>
```

Contoh implementasi menggunakan parameter metaclass:

```
class Metanih(type):
    def __new__(cls, namakelas, basis, dictattr):
        a = super().__new__(cls, namakelas, basis, dictattr)
        a.jenis_metaclass = "Ini metaclass lhoooo"
        return a

class Misal(metaclass=Metanih):
    pass

contoh = Misal()
print(contoh.jenis_metaclass)
```

```
input
Ini metaclass lhoooo
```