TUGAS PRAKTIKUM 6

PEMROGRAMAN BERORIENTASI OBJEK

Disusun Oleh :

Heni Artha Uli br Turnip 121140080

PROGRAM STUDI TEKNIK INFORMATIKA

JURUSAN TEKNIK ELEKTRO, INFORMATIKA DAN SISTEM FISIS

INSTITUT TEKNOLOGI SUMATERA

2023

DAFTAR ISI

BAB I
RESUME

A. Abstract Class

Kelas abstrak, juga dikenal sebagai abstract class, merupakan kelas yang berada di puncak hierarki class. Kelas ini tidak dapat diinstansiasi karena bersifat abstrak. Abstract class hanya berisi variabel umum dan deklarasi method tanpa implementasi detailnya (abstract method). Implementasi detail method tersebut ditentukan oleh kelas-kelas turunannya. Kelas turunan dari abstract class wajib mengimplementasikan semua method abstrak yang ada di dalam abstract class tersebut, sehingga abstract class menjadi kerangka kerja atau blueprint bagi kelas-kelas turunannya. Abstract class sering digunakan dalam pemrograman berorientasi objek untuk membuat model atau template yang dapat digunakan oleh banyak kelas turunannya. Abstract class dapat dianggap sebagai templat untuk kelas lain. Hal ini memungkinkan kita untuk menentukan kumpulan method yang harus diimplementasikan di kelas anak yang berasal dari abstract class. Kelas abstrak adalah kelas yang memiliki satu atau lebih method abstrak. Method yang memiliki deklarasi tetapi tidak memiliki implementasi disebut abstrak. Abstract class digunakan untuk membangun unit fungsional yang besar. Abstract class juga digunakan ketika ingin menyediakan antarmuka umum untuk berbagai implementasi komponen.

Dalam Python, modul yang digunakan untuk menggunakan Abstract Base Classes (ABC) adalah ABC. Fungsi abstrak dalam abstract class ditandai dengan decorator @abstractmethod pada fungsi yang dibuat.

Berikut adalah contoh implementasi abstract class pada Python:

```python
from abc import ABC, abstractmethod

class AbstractClassExample(ABC):

    @abstractmethod
    def method1(self):
        pass

    @abstractmethod
    def method2(self):
        pass

class ConcreteClassExample(AbstractClassExample):

    def method1(self):
        # Implementasi method1
        pass

    def method2(self):
        # Implementasi method2
        pass
```

Dalam contoh di atas, AbstractClassExample adalah abstract class yang memiliki dua abstract method: method1 dan method2. Kelas ConcreteClassExample merupakan turunan dari AbstractClassExample dan mengimplementasikan kedua method abstrak tersebut.

Penting untuk dicatat bahwa jika kita tidak mengimplementasikan semua method abstrak yang ada di abstract class saat membuat turunannya, maka akan terjadi error pada saat pembuatan objek dari turunan tersebut.

B. Interface

Interface dalam Python biasanya digunakan untuk mendefinisikan kontrak dan menentukan apa yang dapat dilakukan oleh sebuah kelas, tanpa memperhatikan implementasinya. Interface berfungsi sebagai template untuk desain kelas. Interface, seperti kelas, menentukan method, tetapi method-method ini bersifat abstrak, yang berarti mereka hanya didefinisikan oleh interface tanpa implementasi. Interface memungkinkan kelas-kelas lain untuk mengimplementasikan method-method ini dan memberikan makna konkret kepada fungsi abstrak dalam interface.

Pendekatan Python terhadap desain interface berbeda dari bahasa lain seperti Java, Go, dan C++. Bahasa-bahasa ini memiliki kata kunci "interface" untuk mendefinisikan interface, tetapi Python tidak memiliki kata kunci tersebut. Selain itu, dalam Python, kelas yang mengimplementasikan sebuah interface tidak perlu mendefinisikan semua method abstrak yang ada dalam interface tersebut.

Berikut adalah contoh implementasi interface pada Python:

```python
from abc import ABC, abstractmethod

class MyInterface(ABC):

    @abstractmethod
    def method1(self):
        pass

    @abstractmethod
    def method2(self):
        pass

class MyClass(MyInterface):

    def method1(self):
        # Implementasi method1
        pass

    def method2(self):
        # Implementasi method2
        pass
```

Dalam contoh di atas, MyInterface merupakan interface yang memiliki dua method abstrak: method1 dan method2. Kelas MyClass mengimplementasikan interface MyInterface dan memberikan implementasi nyata untuk kedua method abstrak tersebut.

Perbedaan antara interface dan abstract class adalah sebagai berikut:

a. Interface:

- Secara default, interface adalah publik dan tidak mendukung access specifier.
- Interface hanya berisi tanda tangan method dan tidak memiliki implementasi.
- Prosesnya relatif lebih lambat.
- Tidak memiliki field (variabel anggota).
- Interface hanya digunakan untuk mengimplementasikan method abstrak.

b. Abstract Class:

- Abstract class dapat memiliki access specifier.
- Abstract class dapat memiliki implementasi method secara bertahap.
- Prosesnya relatif lebih cepat.
- Dapat memiliki field dan konstanta.
- Dapat menampung method non-abstrak.

1. Informal Interface

Dalam beberapa situasi, batasan yang ketat dari formal interface Python mungkin tidak diperlukan. Karena sifat dinamis Python, kita dapat membuat interface yang lebih tidak resmi. Interface informal dalam Python dapat dianggap sebagai kelas yang mendefinisikan method-method yang dapat diganti tanpa memerlukan penerapan yang ketat. Dalam pemrograman, interface informal adalah pola atau konvensi yang digunakan oleh para programmer untuk mengimplementasikan fungsionalitas serupa di berbagai kelas tanpa menggunakan fitur formal interface yang ada dalam bahasa pemrograman.

Dalam interface informal, para programmer setuju untuk menggunakan method-method dengan nama dan tipe parameter yang sama pada kelas-kelas yang berbeda sebagai interface. Dengan kata lain, method dengan nama dan tipe parameter yang sama pada kelas-kelas yang berbeda dianggap sebagai interface secara informal. Sebagai contoh, dalam bahasa pemrograman Python, tidak ada fitur formal interface seperti yang ada dalam bahasa Java. Namun, kita dapat menggunakan interface informal dengan menggunakan method-method yang memiliki nama dan tipe parameter yang sama pada kelas-kelas yang berbeda.

Penggunaan interface informal sering memudahkan para programmer dalam membuat kelas-kelas yang berhubungan dan dapat saling berinteraksi. Namun, penggunaan interface informal juga dapat membingungkan jika tidak ada dokumentasi atau konvensi yang jelas dalam penggunaannya.

Berikut adalah contoh implementasi informal interface pada Python:

```python
class InformalInterface1:
    def method1(self):
        pass

    def method2(self, parameter):
        pass

class InformalInterface2:
    def method1(self):
        pass

    def method2(self, parameter):
        pass

class MyClass1(InformalInterface1):
    def method1(self):
        # Implementasi method1
        pass

    def method2(self, parameter):
        # Implementasi method2
        pass

class MyClass2(InformalInterface2):
    def method1(self):
        # Implementasi method1
        pass

    def method2(self, parameter):
        # Implementasi method2
        pass
```

Dalam contoh di atas, InformalInterface1 dan InformalInterface2 adalah dua interface informal yang didefinisikan dengan menggunakan method-method yang memiliki nama dan tipe parameter yang sama. Kelas MyClass1 dan MyClass2 mengimplementasikan interface-interface tersebut dengan memberikan implementasi konkret untuk method-method yang sesuai.

Perlu dicatat bahwa dalam penggunaan interface informal, tidak ada pemeriksaan statis yang dilakukan oleh bahasa pemrograman untuk memastikan bahwa kelas-kelas yang mengimplementasikan interface tersebut memenuhi kontraknya. Oleh karena itu, dokumentasi yang jelas dan konvensi yang disepakati penting dalam penggunaan interface informal.

2. Formal Interface

Formal interface dalam bahasa pemrograman Python adalah sebuah konstruksi bahasa yang memungkinkan para programmer untuk menentukan sekumpulan method yang harus diimplementasikan oleh kelas-kelas yang menggunakan interface tersebut. Tujuan dari formal interface adalah untuk memastikan bahwa sebuah kelas yang mengimplementasikan interface tersebut memiliki method-method yang diperlukan untuk berinteraksi dengan kelas-kelas lain. Dalam Python, formal interface dapat diimplementasikan menggunakan modul abc (Abstract Base Class). Modul ini memungkinkan para programmer untuk

membuat kelas-kelas abstrak yang berfungsi sebagai formal interface. Sebuah kelas abstrak adalah kelas yang memiliki satu atau lebih method abstrak, yaitu method yang didefinisikan tetapi tidak diimplementasikan di kelas tersebut.

Formal interface adalah konsep yang ditegakkan secara eksplisit. Untuk membuat sebuah formal interface, kita perlu menggunakan ABC (Abstract Base Class). ABC mengajarkan kita untuk mendefinisikan kelas abstrak dan mendefinisikan method-method pada kelas dasar sebagai method-method abstrak, sehingga setiap objek yang diturunkan dari kelas dasar tersebut harus mengimplementasikan method-method tersebut. Untuk mengimplementasikan sebuah formal interface, sebuah kelas harus melakukan pewarisan (inheritance) dari kelas abstrak dan mengimplementasikan method-method abstrak yang telah didefinisikan dalam kelas abstrak tersebut. Jika sebuah kelas tidak mengimplementasikan semua method yang diperlukan dalam kelas abstrak, maka kelas tersebut tidak dapat diinstansiasi.

Berikut adalah contoh implementasi formal interface dalam Python:

```python
from abc import ABC, abstractmethod

class FormalInterface(ABC):

    @abstractmethod
    def method1(self):
        pass

    @abstractmethod
    def method2(self, parameter):
        pass

class MyClass(FormalInterface):

    def method1(self):
        # Implementasi method1
        pass

    def method2(self, parameter):
        # Implementasi method2
        pass
```

Dalam contoh di atas, FormalInterface adalah sebuah kelas abstrak yang berfungsi sebagai formal interface. Interface ini memiliki dua method abstrak: method1 dan method2. Kelas MyClass mengimplementasikan formal interface tersebut dengan memberikan implementasi konkret untuk kedua method abstrak tersebut.

Perlu dicatat bahwa dalam penggunaan formal interface, bahasa pemrograman Python akan melakukan pemeriksaan statis untuk memastikan bahwa kelas-kelas yang mengimplementasikan interface tersebut memenuhi semua kontraknya.

C. Metaclass

Pada Python, metaclass adalah sebuah kelas yang mengatur perilaku sebuah kelas. Metaclass ini merupakan turunan dari kelas-kelas lainnya. Dalam Python, kelas menggambarkan cara sebuah instance kelas berperilaku. Untuk memahami konsep metaclass dengan baik, pengalaman sebelumnya dalam bekerja dengan kelas-kelas Python sangat diperlukan. Metaclass digunakan untuk membuat kelas baru dengan perilaku khusus. Metaclass memiliki kemampuan untuk mengontrol pembuatan kelas, seperti mengubah atribut atau metode dari kelas, mengubah cara pewarisan kelas dilakukan, dan sebagainya.

Penggunaan metaclass umumnya terjadi dalam situasi di mana kita ingin membuat kelas dengan perilaku khusus atau melakukan validasi dan verifikasi pada kelas sebelum kelas tersebut dibuat. Contohnya, dalam beberapa framework web Python, metaclass digunakan untuk memastikan bahwa setiap kelas model database memiliki atribut dan metode yang sesuai. Secara default, setiap kelas di Python memiliki metaclass yang merupakan turunan dari kelas type. Namun, kita dapat membuat metaclass kustom dengan membuat sebuah kelas yang mewarisi dari kelas type, dan kemudian menggunakan metaclass tersebut saat membuat kelas baru.

Berikut adalah contoh implementasi metaclass dalam Python:

```
class MyMetaClass(type):
    def __new__(cls, name, bases, attrs):
        # Modifikasi atau validasi attrs
        modified_attrs = ...
        return super().__new__(cls, name, bases, modified_attrs)

class MyClass(metaclass=MyMetaClass):
    # Definisi kelas dengan metaclass kustom
    pass
```

Dalam contoh di atas, MyMetaClass adalah sebuah metaclass kustom yang merupakan turunan dari kelas type. Metaclass ini memiliki metode __new__ yang akan dipanggil saat menciptakan kelas baru. Di dalam metode __new__, kita dapat memodifikasi atau melakukan validasi terhadap atribut-atribut kelas sebelum kelas tersebut dibuat. Kemudian, kita menggunakan metaclass MyMetaClass saat mendefinisikan kelas MyClass.

Dengan menggunakan metaclass, kita dapat mengontrol dan mengubah perilaku kelas sesuai kebutuhan kita, dan melibatkan proses manipulasi atau validasi sebelum pembuatan kelas yang sesuai.

BAB II

KESIMPULAN

Dalam Python, interface digunakan untuk menentukan kontrak dan fungsionalitas yang dapat dilakukan oleh sebuah kelas tanpa memperhatikan implementasinya. Interface berfungsi sebagai template untuk desain kelas, yang mencakup metode-metode yang bersifat abstrak. Metode abstrak ini hanya didefinisikan oleh interface tanpa memberikan implementasinya. Kelas dapat mencapai hal ini dengan mengimplementasikan interface dan memberikan implementasi nyata pada metode-metode abstrak yang ada dalam interface. Interface digunakan dalam berbagai situasi, seperti ketika ingin membuat sebuah kelas atau objek yang dapat digunakan oleh banyak kelas atau objek lainnya, membuat plugin atau modul yang dapat diintegrasikan dengan sistem atau aplikasi lain, dan saat ingin membuat aplikasi yang mudah diuji (testable).

Abstract class, atau kelas abstrak, adalah kelas yang berada di puncak hierarki kelas dan tidak dapat diinstansiasi karena bersifat abstrak. Kelas ini hanya berisi variabel umum dan deklarasi metode tanpa implementasi detail (metode abstrak). Kelas turunan dari abstract class harus mengimplementasikan semua metode abstrak yang ada dalam abstract class tersebut, sehingga abstract class menjadi kerangka kerja atau blueprint bagi kelas-kelas turunannya. Abstract class digunakan saat ingin membuat sebuah abstraksi yang mendefinisikan perilaku umum atau fungsionalitas dari kelas, tanpa memberikan implementasi konkret dari kelas tersebut. Dalam hal ini, hanya ditentukan metode atau atribut yang perlu ada dalam kelas tanpa harus menentukan bagaimana metode atau atribut tersebut harus diimplementasikan.

Perbedaan antara Abstract Class dengan Interface adalah sebagai berikut:

a. Interface
    - Interface secara default bersifat publik dan tidak mendukung access specifier.
    - Hanya berisi tanda tangan metode dan tidak berisi implementasi.
    - Kecepatan proses relatif lebih lambat.
    - Tidak memiliki field.
    - Interface hanya digunakan untuk memperluas metode-metode abstrak.
b. Kelas Abstrak
    - Kelas abstrak dapat memiliki access specifier.
    - Implementasinya dapat dilakukan secara bertahap.
    - Kecepatan proses relatif lebih cepat.
    - Dapat memiliki field dan konstanta.
    - Dapat menampung metode non-abstrak.

Kelas konkret adalah kelas yang memiliki implementasi lengkap untuk semua metode dan properti yang dimilikinya, sehingga dapat langsung digunakan untuk membuat objek. Kelas konkret dibedakan dari kelas abstrak yang hanya memiliki definisi metode dan properti tanpa implementasi lengkap. Kelas abstrak sering digunakan sebagai kerangka dasar untuk kelas konkret yang mengimplementasikan metode dan properti yang diwarisi dari kelas abstrak. Kelas konkret digunakan ketika ingin membuat objek yang spesifik dengan implementasi yang sudah lengkap. Metaclass dalam Python adalah kelas yang menggambarkan perilaku kelas.

BAB III

DAFTAR PUSTAKA

bestharadhakrishna. (2021, Maret 19). *Abstract Classes in Python*. Retrieved from GeeksforGeeks: https://www.geeksforgeeks.org/abstract-classes-in-python/

Muhardian, A. (2019, Desember 28). *Tutorial Java OOP: Memahami Interface di Java (dan Contohnya)*. Retrieved from Petani Kode: https://www.petanikode.com/java-oop-interface/

Mwiti, D. (2018, Desember). *Introduction to Python Metaclasses*. Retrieved from DataCamp: https://www.datacamp.com/tutorial/python-metaclasses

Sturtz, J. (n.d.). *Python Metaclasses*. Retrieved from Real Python: https://realpython.com/pythonmetaclasses/#author