LAPORAN PRAKTIKUM
PEMROGRAMAN BERIOTIENTASI OBJEK (RB)
TUGAS 6

Oleh:

Maharani Triza Putri (121140071)

Program Studi Teknik Informatika

Institut Teknologi Sumatera

2023

ABSTRAKSI

A. Konsep Abstrasksi

Konsep abstraksi merujuk pada proses menyembunyikan kompleksitas dan detail yang tidak relevan dari suatu objek atau sistem. Ini adalah pendekatan yang digunakan dalam pemrograman dan desain untuk menyederhanakan pemahaman dan penggunaan suatu entitas dengan fokus pada aspek yang penting.

Dalam konteks pemrograman, abstraksi melibatkan menciptakan struktur atau antarmuka yang mengisolasi implementasi atau detail internal suatu objek dari pengguna. Hal ini memungkinkan pengguna atau pengembang untuk berinteraksi dengan objek tersebut melalui tingkat abstraksi yang lebih tinggi, tanpa perlu memahami semua detail implementasinya.

Abstraksi memungkinkan kita untuk memperhatikan aspek penting atau fungsionalitas suatu objek atau sistem, sementara kompleksitas dan detail yang tersembunyi diabaikan atau disembunyikan. Ini memberikan manfaat berikut:

1. Kompleksitas yang tersembunyi
   Abstraksi memungkinkan kita untuk menyembunyikan detail implementasi yang rumit atau kompleks dari pengguna. Pengguna hanya perlu tahu cara menggunakan objek atau sistem tersebut dan tidak perlu terlibat secara langsung dengan implementasinya.

2. Pemahaman yang lebih mudah
   Dengan menyederhanakan antarmuka dan menyembunyikan detail yang tidak perlu, abstraksi membuat objek atau sistem lebih mudah dipahami. Pengguna dapat fokus pada fungsionalitas yang relevan dan tidak perlu terbebani oleh kompleksitas internal.

3. Penggunaan yang lebih mudah
   Dengan menyediakan antarmuka yang sederhana dan intuitif, abstraksi membuat penggunaan objek atau sistem lebih mudah. Pengguna hanya perlu memahami cara berinteraksi dengan antarmuka yang disediakan tanpa harus memahami semua detail implementasi.

4. Peningkatan fleksibilitas
   Abstraksi memungkinkan perubahan implementasi di belakang antarmuka tanpa mempengaruhi pengguna. Jika implementasi diubah atau ditingkatkan, pengguna tetap dapat menggunakan objek atau sistem tersebut tanpa mengubah cara mereka berinteraksi.

5. Modularitas
   Abstraksi memungkinkan pemisahan fungsionalitas ke dalam komponen yang terpisah. Komponen-komponen ini dapat dianggap sebagai "blok bangunan" yang dapat digunakan ulang dan dikombinasikan untuk membuat solusi yang lebih kompleks.

Abstraksi adalah konsep inti dalam pemrograman berorientasi objek (OOP), di mana objek dan kelas digunakan untuk menerapkan abstraksi. Dalam OOP, kelas menyediakan abstraksi melalui antarmuka publik yang mendefinisikan operasi yang dapat dilakukan pada objek tersebut, sementara detail implementasi tersembunyi di dalam kelas tersebut.

Secara keseluruhan, abstraksi adalah konsep yang penting dalam pemrograman dan desain, karena memungkinkan kita untuk menghadapi kompleksitas dengan cara yang terorganisir, memfasilitasi pemahaman yang lebih baik, penggunaan yang lebih mudah, dan fleksibilitas yang lebih besar.

B. Kelas Abstrak dalam Python

Kelas Abstrak dalam Python merujuk pada kelas yang tidak dapat diinstansiasi langsung dan berfungsi sebagai kerangka kerja untuk kelas-kelas turunannya. Kelas ini berisi metode-metode abstrak yang harus diimplementasikan oleh setiap subkelas yang mewarisinya.

Dalam Python, kelas abstrak didefinisikan menggunakan modul ABC (Abstract Base Classes). Modul ini menyediakan mekanisme untuk mendefinisikan kelas abstrak dan metode abstrak. Untuk membuat kelas abstrak, kita perlu melakukan hal berikut:

1. Mengimpor modul ABC dari modul abc.
2. Mendefinisikan kelas abstrak dengan mewarisi kelas ABC dari modul abc.

Contoh sintaks untuk mendefinisikan kelas abstrak dalam Python:

```
from abc import ABC, abstractmethod

class AbstractClass(ABC):

    @abstractmethod
    def abstract_method(self):
        pass
```

Dalam contoh di atas, kita mengimpor modul ABC dan dekorator abstractmethod dari modul abc. Kemudian, kita mendefinisikan kelas AbstractClass yang mewarisi kelas ABC dari modul abc.

Di dalam kelas AbstractClass, kita mendeklarasikan metode abstract_method sebagai metode abstrak dengan menggunakan dekorator @abstractmethod. Metode ini tidak memiliki implementasi konkret dan harus diimplementasikan oleh setiap subkelas yang mewarisi kelas ini.

Kelas abstrak tidak dapat diinstansiasi langsung. Namun, kita dapat membuat subkelas yang mewarisi kelas abstrak dan memberikan implementasi konkret untuk metode-metode abstrak yang ditentukan oleh kelas abstrak tersebut.

Contoh implementasi subkelas dari kelas abstrak:

```
class ConcreteClass(AbstractClass):

    def abstract_method(self):
        print("Implementasi metode abstrak di subkelas")
```

Dalam contoh di atas, kita membuat subkelas ConcreteClass yang mewarisi AbstractClass dan memberikan implementasi konkret untuk metode abstract_method.

Kelas abstrak dan metode abstrak memberikan struktur dan kontrak yang jelas bagi kelas-kelas turunannya. Mereka memastikan bahwa setiap subkelas harus mengimplementasikan metode-metode abstrak yang ditentukan oleh kelas abstrak. Dengan menggunakan kelas abstrak, kita dapat menciptakan hierarki kelas yang fleksibel dengan fungsionalitas yang terdefinisi dengan jelas.

C. Implementasi Kelas Abstrak dengan Modul ABC

Implementasi Kelas Abstrak dengan Modul ABC dalam Python melibatkan penggunaan modul ABC (Abstract Base Classes) untuk mendefinisikan kelas abstrak dan metode abstrak. Modul ABC menyediakan dekorator abstractmethod yang digunakan untuk menandai metode sebagai metode abstrak yang harus diimplementasikan oleh subkelasnya.

Berikut adalah langkah-langkah untuk mengimplementasikan kelas abstrak menggunakan modul ABC:

1. Import modul ABC dan abstractmethod dari modul abc:

```
from abc import ABC, abstractmethod
```

2. Buat kelas abstrak dengan mewarisi kelas ABC:

```
class AbstractClass(ABC):
    pass
```

Dalam contoh di atas, kita mendefinisikan kelas AbstractClass yang mewarisi kelas ABC dari modul abc. Kelas ini akan menjadi kelas abstrak yang tidak dapat diinstansiasi secara langsung.

3. Tandai metode sebagai metode abstrak dengan menggunakan dekorator abstractmethod:

```
class AbstractClass(ABC):

    @abstractmethod
    def abstract_method(self):
        pass
```

Di dalam kelas abstrak, kita mendefinisikan metode abstract_method dan menandainya sebagai metode abstrak menggunakan dekorator @abstractmethod. Metode ini tidak memiliki implementasi konkret dan harus diimplementasikan oleh setiap subkelas yang mewarisi kelas ini.

4. Buat subkelas yang mewarisi kelas abstrak dan implementasikan metode abstrak:

```
class ConcreteClass(AbstractClass):

    def abstract_method(self):
        print("Implementasi metode abstrak di subkelas")
```

Dalam contoh di atas, kita membuat subkelas ConcreteClass yang mewarisi AbstractClass dan memberikan implementasi konkret untuk metode abstract_method.

Dengan langkah-langkah di atas, kita telah mengimplementasikan kelas abstrak dengan modul ABC. Kelas abstrak menyediakan kerangka kerja yang jelas dan kontrak yang harus

diikuti oleh setiap subkelasnya. Subkelas harus memberikan implementasi konkret untuk metode-metode abstrak yang ditentukan oleh kelas abstrak.

Perlu diingat bahwa jika sebuah kelas memiliki setidaknya satu metode abstrak, maka kelas tersebut juga harus dideklarasikan sebagai kelas abstrak dengan mewarisi kelas ABC. Kelas abstrak tidak dapat diinstansiasi langsung, tetapi digunakan sebagai kerangka kerja untuk membuat subkelas yang mengimplementasikan metode-metode abstrak tersebut.

INTERFACE

Interface merupakan sebuah konsep dalam pemrograman yang menggambarkan sebuah kontrak atau spesifikasi yang diberikan kepada suatu objek. Interface mendefinisikan metode-metode yang harus diimplementasikan oleh suatu kelas, tetapi tidak memberikan detail implementasi konkrit dari metode-metode tersebut. Dengan kata lain, interface menyediakan sebuah antarmuka yang menentukan perilaku yang diharapkan dari suatu objek, tetapi tidak peduli dengan bagaimana objek tersebut mencapai perilaku tersebut.

Interface menyediakan abstraksi tingkat tinggi yang memisahkan definisi dan implementasi. Dalam sebuah interface, hanya deklarasi metode dan konstanta yang didefinisikan, tanpa menyertakan detail implementasi. Objek yang mengimplementasikan suatu interface diharuskan untuk menyediakan implementasi konkret untuk setiap metode yang dideklarasikan dalam interface tersebut.

Beberapa hal penting yang perlu diketahui tentang interface:

1. Interface hanya berisi deklarasi metode dan konstanta, tanpa implementasi konkret.

2. Interface tidak dapat diinstansiasi langsung, tetapi digunakan sebagai kontrak untuk kelas-kelas yang mengimplementasikannya.

3. Setiap kelas yang mengimplementasikan interface harus memberikan implementasi konkret untuk semua metode yang dideklarasikan dalam interface tersebut.

4. Interface dapat digunakan untuk mencapai polimorfisme, di mana objek dari berbagai kelas yang mengimplementasikan interface dapat diperlakukan secara seragam melalui antarmuka yang sama.

Dalam Python, tidak ada konsep interface yang sama persis seperti dalam bahasa pemrograman lain seperti Java atau C#. Namun, dengan menggunakan modul ABC (Abstract Base Classes) dalam Python, kita dapat menciptakan kelas abstrak yang berfungsi sebagai pengganti interface. Kelas abstrak di Python dapat memiliki metode abstrak yang harus diimplementasikan oleh kelas-kelas turunannya, mirip dengan kontrak yang diberikan oleh interface.

Pada dasarnya, interface adalah kontrak yang mendefinisikan perilaku yang diharapkan dari suatu objek, tetapi tidak mengatur implementasi konkrit. Hal ini memungkinkan pemisahan antara

definisi dan implementasi, dan memungkinkan penggunaan polimorfisme dan abstraksi tingkat tinggi dalam pemrograman.

KESIMPULAN

1. Interface adalah spesifikasi yang menjelaskan fungsi-fungsi yang harus ada dalam suatu kelas tanpa memberikan implementasi konkret dari fungsi tersebut. Interface diperlukan untuk memastikan bahwa kelas-kelas yang mengimplementasikannya memiliki implementasi dari fungsi-fungsi tersebut.

2. Kelas abstrak adalah kelas yang tidak dapat diinstansiasi sendiri, tetapi berfungsi sebagai basis atau kerangka untuk mewarisi kelas-kelas lain. Kelas abstrak sering berisi fungsi-fungsi umum yang dapat digunakan kembali oleh kelas-kelas turunannya. Kelas abstrak digunakan ketika kita ingin mendefinisikan kelas dasar yang menyediakan fungsi-fungsi umum tetapi tidak lengkap atau tidak sepenuhnya diimplementasikan. Ini berbeda dengan interface yang digunakan ketika terdapat fitur yang ingin diimplementasikan.

3. Kelas konkret adalah kelas yang dapat diinstansiasi secara langsung dan menyediakan implementasi lengkap untuk semua metodenya. Kelas konkret tidak mengandung metode abstrak seperti yang terdapat dalam kelas abstrak. Kita menggunakan kelas konkret ketika kita ingin membuat objek yang semua metodenya telah diimplementasikan sepenuhnya dan siap digunakan.

4. Metaclass adalah kelas yang mendefinisikan perilaku kelas lain. Metaclass bisa dianggap sebagai kelas dari kelas. Ketika kita membuat kelas baru, kita dapat menentukan metaclass yang akan menentukan bagaimana kelas baru tersebut dibuat dan perilakunya. Metaclass digunakan ketika kita ingin menyesuaikan perilaku kelas di luar kemampuan yang dapat dicapai melalui pewarisan biasa. Metaclass berbeda dari pewarisan biasa karena metaclass menentukan perilaku kelas dan bagaimana kelas-kelas itu dibuat, sedangkan pewarisan hanya menentukan hubungan antara kelas dan kelas induknya.