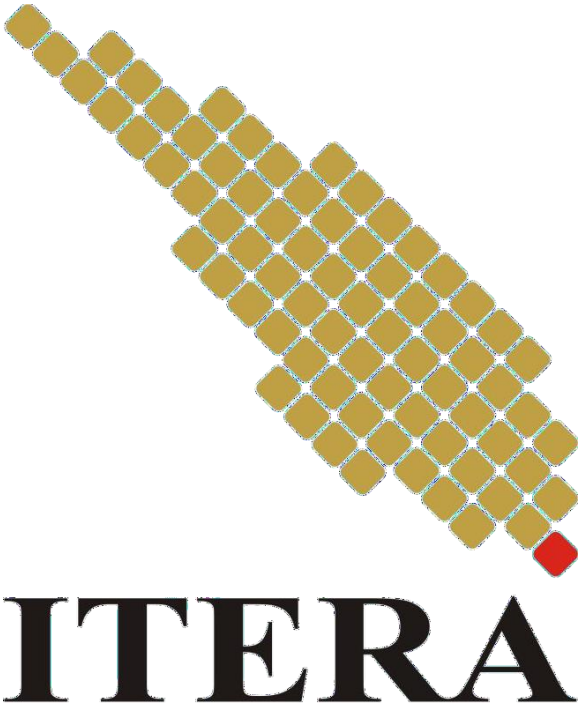
# LAPORAN PRAKTIKUM

# PEMROGRAMAN BERORIENTASI OBJEK (RD)

# MODUL 6

Oleh :

Lilis Swastika (121140233)




Program Studi Teknik Informatika

Institut Teknologi Sumatera

2023

# Daftar Isi

Daftar Isi ..... 2
Ringkasan ..... 3
1.1 Kelas Abstrak ..... 3
1.2 Interface ..... 4
1.2.1 Informal Interface ..... 4
1.2.2 Formal Interface ..... 4
1.2.3 Metaclass ..... 5
Kesimpulan ..... 7
Daftar Pustaka ..... 8

## Ringkasan

### 1.1 Kelas Abstrak

Sebuah kelas disebut kelas abstrak jika berisi satu atau lebih metode abstrak. Metode abstrak adalah metode yang dideklarasikan tetapi tidak mengandung implementasi. Kelas abstrak tidak boleh dipakai dan metode abstraknya harus diimplementasikan oleh subkelasnya.

Kelas dasar abstrak menyediakan cara untuk mendefinisikan antarmuka ketika teknik lain seperti hasattr() akan menjadi kikuk atau sesuatu yang salah (misalnya menggunakan metode ajaib). ABC memperkenalkan subclass virtual, i. H. Kelas yang tidak mewarisi dari kelas tetapi masih dikenali oleh fungsi isinstance() dan issubclass(). Banyak ABC dibangun ke dalam Python. ABC untuk struktur data seperti iterator, generator, set, pemetaan, dll. Didefinisikan dalam modul collections.abc. Modul angka mendefinisikan menara angka, yang merupakan kumpulan kelas dasar untuk tipe data angka. Modul 'abc' di pustaka Python menyediakan infrastruktur untuk mendefinisikan kelas dasar abstrak Anda sendiri. Anda dapat memanggil atribut pada kelas konkret dengan mendefinisikannya menggunakan @abstractproperty.

Kelas abstrak tidak lengkap karena mereka memiliki metode yang tidak dimiliki orang lain. Jika Python memungkinkan pemrogram membuat objek untuk kelas abstrak, gunakan objek itu jika seseorang memanggil metode abstrak tetapi implementasi nyata tidak memanggilnya. Itu sebabnya pemrogram menggunakan kelas abstrak sebagai templat dan jika perlu kami memperluas dan mengonfigurasinya sebelum menggunakannya. Kelas abstrak tidak dapat dipakai karena itu bukan kelas konkret. Saat pemrogram membuat objek untuk kelas abstrak, itu akan menimbulkan kesalahan.

Contoh codingan:

```python
import abc

class Shape(metaclass=abc.ABCMeta):
    @abc.abstractmethod
    def area(self):
        pass

class Rectangle(Shape):
    def __init__(self, width, height):
        self.width = width
        self.height = height

    def area(self):
        return self.width * self.height

class Circle(Shape):
    def __init__(self, radius):
        self.radius = radius

    def area(self):
        return 3.14 * self.radius ** 2

r = Rectangle(5, 10)
print(r.area()) # Output: 50

c = Circle(7)
print(c.area()) # Output: 153.86

```

Dalam contoh di atas, ***“Shape”*** adalah kelas abstrak yang memiliki satu metode abstrak yaitu ***“area()”***. Kelas “***Rectangle”*** dan “***Circle”*** mewarisi kelas ***“Shape”*** dan mengimplementasikan metode ***“area()”*** yang didefinisikan sebagai metode konkret.

## 1.2 Interface

Interface memainkan peran penting dalam rekayasa perangkat lunak. Seiring berkembangnya aplikasi, semakin sulit untuk mengelola pembaruan dan perubahan pada basis kode. Pada tingkat tinggi, antarmuka pengguna berfungsi sebagai cetak biru untuk desain pelajaran. Seperti kelas, antarmuka menentukan metode. Tidak seperti kelas, metode bersifat abstrak. Metode abstrak adalah metode yang didefinisikan oleh antarmuka. Itu tidak menerapkan metode. Ini dilakukan oleh kelas-kelas, yang kemudian mengimplementasikan antarmuka dan memberi arti konkret pada metode abstrak antarmuka.

### 1.2.1 Informal Interface

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.

Contoh codingan:

```python
class PaymentGateway:
    def process_payment(self, amount):
        raise NotImplementedError("process_payment() harus diimplementasikan di kelas turunan")

class CreditCardGateway(PaymentGateway):
    def process_payment(self, amount):
        print(f"Membuat pembayaran sebesar {amount} dengan kartu kredit")

class BankTransferGateway(PaymentGateway):
    def process_payment(self, amount):
        print(f"Membuat pembayaran sebesar {amount} melalui transfer bank")

class PaymentProcessor:
    def __init__(self, gateway: PaymentGateway):
        self.gateway = gateway

    def process(self, amount):
        self.gateway.process_payment(amount)

cc_gateway = CreditCardGateway()
payment_processor = PaymentProcessor(cc_gateway)
payment_processor.process(100)

bt_gateway = BankTransferGateway()
payment_processor = PaymentProcessor(bt_gateway)
payment_processor.process(200)
```

### 1.2.2 Formal Interface

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.

Contoh codingan:

```python
import abc

class PaymentGateway(metaclass=abc.ABCMeta):
    @abc.abstractmethod
    def process_payment(self, amount):
        pass

class CreditCardGateway(PaymentGateway):
    def process_payment(self, amount):
        print(f"Membuat pembayaran sebesar {amount} dengan kartu kredit")

class BankTransferGateway(PaymentGateway):
    def process_payment(self, amount):
        print(f"Membuat pembayaran sebesar {amount} melalui transfer bank")

class PaymentProcessor:
    def __init__(self, gateway: PaymentGateway):
        self.gateway = gateway

    def process(self, amount):
        self.gateway.process_payment(amount)

cc_gateway = CreditCardGateway()
payment_processor = PaymentProcessor(cc_gateway)
payment_processor.process(100)

bt_gateway = BankTransferGateway()
payment_processor = PaymentProcessor(bt_gateway)
payment_processor.process(200)
```

Graphical User Interface (GUI) baru-baru ini menjadi populer untuk membuat antarmuka pengguna, menampilkan berbagai elemen grafis seperti menu, ikon, dan jendela. GUI adalah antarmuka yang sering beroperasi di PC, tablet, dan perangkat akhir lainnya.

GUI menggunakan elemen grafik seperti ikon, menu, dan gambar untuk membuatnya lebih mudah digunakan oleh pengguna manusia. Antarmuka pengguna grafis digunakan untuk setiap aplikasi. Hampir semua perangkat lunak pengguna akhir saat ini dilengkapi dengan GUI.

### 1.2.3 Metaclass

Metaclass di Python adalah kelas dari kelas yang menentukan bagaimana kelas itu berperilaku. Kelas itu sendiri adalah turunan dari metaclass. Kelas dalam Python menentukan bagaimana keadaan kelas berperilaku.

Contoh codingan:

```python
class MyMetaClass(type):
    def __new__(cls, name, bases, attrs):
        print(f"Membuat kelas baru dengan nama {name}")
        return super().__new__(cls, name, bases, attrs)

class MyClass(metaclass=MyMetaClass):
    pass
```

Dalam contoh di atas, ***"MyMetaClass"*** adalah sebuah kelas yang menjadi metaclass. Ketika kita membuat kelas baru ***"MyClass"*** dan menentukan metaclass-nya sebagai ***"MyMetaClass",*** maka metode ***"_new_()"*** dari ***"MyMetaClass"*** akan dipanggil saat pembuatan objek kelas baru.

Metode ***“ new ()”*** dari ***“MyMetaClass”*** menerima tiga argumen, yaitu ***“cls”, “name”, “bases”***, dan ***“attrs”. “cls”*** adalah kelas ***“MyMetaClass”*** itu sendiri, name adalah nama kelas yang akan dibuat, ***“bases”*** adalah kelas-kelas induk dari kelas yang akan dibuat, dan ***“attrs”*** adalah atribut-atribut kelas yang akan dibuat.

Dalam contoh di atas, metode ***“_new_()”*** hanya mencetak pesan untuk menunjukkan bahwa kelas baru telah dibuat. Namun, kita dapat melakukan manipulasi lebih lanjut pada kelas yang baru dibuat dalam metode ***“_new_()”***.

## Kesimpulan

Sebuah kelas disebut kelas abstrak jika berisi satu atau lebih metode abstrak. Metode abstrak adalah metode yang dideklarasikan tetapi tidak mengandung implementasi. Kelas abstrak pada Python digunakan ketika kita ingin membuat sebuah antarmuka yang harus diimplementasikan oleh kelas-kelas turunan, namun tidak ingin menentukan implementasi konkret untuk metode-metode di dalam antarmuka tersebut. Kelas abstrak memungkinkan kita untuk menentukan kontrak antara kelas-kelas yang berbeda agar dapat digunakan secara serupa.

Interface pada Python adalah sekumpulan spesifikasi metode dan properti yang dapat digunakan untuk berkomunikasi dengan objek atau kelas tertentu. Kita dapat menggunakan kelas abstrak pada Python untuk membuat interface. Kelas abstrak adalah kelas yang tidak dapat diinisialisasi, tetapi dapat diwarisi oleh kelas-kelas turunan. Kelas abstrak dapat memiliki metode abstrak, yang harus diimplementasikan oleh kelas-kelas turunan. Kita perlu menggunakan interface pada Python ketika kita ingin membuat sebuah antarmuka yang harus diimplementasikan oleh kelas-kelas turunan, namun tidak ingin menentukan implementasi konkret untuk metode-metode di dalam antarmuka tersebut. Interface memungkinkan kita untuk menentukan kontrak antara kelas-kelas yang berbeda agar dapat digunakan secara serupa.

Interface pada Python adalah sebuah kontrak yang harus dipenuhi oleh kelas lain dan tidak memiliki implementasi, sedangkan class abstrak adalah sebuah kerangka kerja untuk kelas lain yang dapat memiliki implementasi dan digunakan sebagai template.

Kelas konkrit pada Python adalah kelas yang dapat diinisialisasi dan langsung digunakan untuk membuat objek. Kelas konkrit memungkinkan kita untuk membuat implementasi konkret dari sebuah kelas dan dapat langsung digunakan tanpa perlu mewarisi kelas lain atau mengimplementasikan sebuah antarmuka. Kita perlu menggunakan kelas konkrit pada Python ketika kita ingin membuat objek dengan implementasi konkret dari sebuah kelas. Kita dapat membuat kelas konkrit untuk berbagai macam tujuan, seperti untuk merepresentasikan objek dalam program, mengolah data, atau menjalankan fungsi tertentu.

Metaclass di Python adalah kelas dari kelas yang menentukan bagaimana kelas itu berperilaku. Kelas itu sendiri adalah turunan dari metaclass. Kita perlu menggunakan metaclass pada Python ketika kita ingin mengubah perilaku atau membuat kelas baru dengan aturan tertentu.

Perbedaan utama antara metaclass dan inheritance pada Python adalah pada level kelas dan objek. Inheritance berlaku pada level objek, di mana kelas turunan atau subclass mewarisi properti dan metode dari kelas induk atau superclass. Sedangkan metaclass berlaku pada level kelas, di mana metaclass digunakan untuk membuat kelas baru dengan aturan yang berbeda dari kelas pada umumnya.

## Daftar Pustaka


APPKEY. (2021, November 17). *Panduan Membuat Graphical User Interface Menggunakan Python*. Retrieved April 9, 2023, from appkey: https://appkey.id/pembuatan-website/front-end/graphical-user-interface/

bestharadhakrishna. (2021, Maret 19). *Abstract Classes in Python*. Retrieved April 9, 2023, from geeksforgeeks: https://www.geeksforgeeks.org/abstract-classes-in-python/

DATACAMP. (2018, December). *Introduction to Python Metaclasses*. Retrieved April 9, 2023, from datacamp: https://www.datacamp.com/tutorial/python-metaclasses#rdl

John, G. (2019, July 30). *Abstract Base Classes in Python (abc)*. Retrieved April 9, 2023, from tutorialspoint: https://www.tutorialspoint.com/abstract-base-classes-in-python-abc

Murphy, W. (n.d.). *Implementing an Interface in Python*. Retrieved April 9, 2023, from realpython: https://realpython.com/python-interface/