

**RESUME MODUL
LAPORAN PRAKTIKUM
PEMROGRAMAN BERORIENTASI OBJEK**



Disusun oleh:

Nama : Ahmad Rizki Maulana
NIM : 121140105
Kelas : PBO RB

**PROGRAM STUDI TEKNIK INFORMATIKA
INSTITUT TEKNOLOGI SUMATERA
LAMPUNG SELATAN**

BAB I

RESUME

A. Kelas Abstrak

Kelas abstrak adalah kelas yang mempunyai setidaknya satu abstract method, kelas abstrak sendiri ditandai dengan mewarisi kelas ABC (Abstract Base Class). Method didalam kelas abstrak yang tidak mempunyai implementasi dinamakan method abstract. Abstract method adalah method yang tidak memiliki body (hanya deklarasi method), Contohnya : `public abstract void someMethod();`. Kelas abstrak tidak bisa dibuat objeknya, maka dari itu suatu kelas harus dapat diturunkan dimana pada subclass tersebut berisi implementasi dari abstract method yang ada di super class-nya.

Berikut adalah contoh sederhana pembuatan kelas abstraksi :

```
main.py
1 import abc
2
3 class BangunDatar(abc.ABC):
4     @abc.abstractmethod
5     def luas(self):
6         pass
7
8     @abc.abstractmethod
9     def keliling(self):
10        pass
11
12 class Persegi(BangunDatar):
13     def __init__(self, sisi):
14         self.sisi = sisi
15
16     def luas(self):
17         return self.sisi ** 2
18
19     def keliling(self):
20         return 4 * self.sisi
21
22 class Lingkaran(BangunDatar):
23     def __init__(self, jari_jari):
24         self.jari_jari = jari_jari
25
26     def luas(self):
27         return 3.14 * (self.jari_jari ** 2)
28
29     def keliling(self):
30         return 2 * 3.14 * self.jari_jari
31
32 # membuat objek dari kelas Persegi dan Lingkaran
33 persegi = Persegi(5)
34 lingkaran = Lingkaran(7)
35
36 # memanggil metode Luas dan keliling pada objek persegi dan lingkaran
37 print("Luas persegi:", persegi.luas())
38 print("Keliling persegi:", persegi.keliling())
39 print("Luas lingkaran:", lingkaran.luas())
40 print("Keliling lingkaran:", lingkaran.keliling())
```

```
main.py
13 class Persegi(BangunDatar):
14     def __init__(self, sisi):
15         self.sisi = sisi
16
17     def luas(self):
18         return self.sisi ** 2
19
20     def keliling(self):
21         return 4 * self.sisi
22
23 class Lingkaran(BangunDatar):
24     def __init__(self, jari_jari):
25         self.jari_jari = jari_jari
26
27     def luas(self):
28         return 3.14 * (self.jari_jari ** 2)
29
30     def keliling(self):
31         return 2 * 3.14 * self.jari_jari
32
33 # membuat objek dari kelas Persegi dan Lingkaran
34 persegi = Persegi(5)
35 lingkaran = Lingkaran(7)
36
37 # memanggil metode Luas dan keliling pada objek persegi dan lingkaran
38 print("Luas persegi:", persegi.luas())
39 print("Keliling persegi:", persegi.keliling())
40 print("Luas lingkaran:", lingkaran.luas())
41 print("Keliling lingkaran:", lingkaran.keliling())
```

```
Luas persegi: 25
Keliling persegi: 20
Luas lingkaran: 153.86
Keliling lingkaran: 43.96
...Program finished with exit code 0
Press ENTER to exit console.
```

Dalam contoh di atas, terdapat kelas abstraksi BangunDatar yang memiliki dua metode abstrak yaitu luas() dan keliling(). Kelas Persegi dan Lingkaran merupakan turunan dari kelas BangunDatar, sehingga harus mengimplementasikan kedua metode abstrak tersebut.

B. Interface

Interface merupakan kumpulan deklarasi fungsi tanpa implementasi yang mendefinisikan aturan pemanggilan fungsi oleh sembarang class lain. Interface berisi method kosong dan konstanta, dan didalamnya tidak mempunyai statement sehingga deklarasi method dalam interface sama dengan deklarasi abstract method pada abstract class. Public dan abstract merupakan method yang dideklarasikan oleh interface, dan variable yang digunakan adalah public, static, dan final. Berikut merupakan contoh pengimplementasian konsep interface :

```
main.py
1 from abc import ABC, abstractmethod
2
3 class Shape(ABC):
4     @abstractmethod
5     def area(self):
6         pass
7
8     @abstractmethod
9     def perimeter(self):
10        pass
11
12 class Rectangle(Shape):
13     def __init__(self, width, height):
14         self.width = width
15         self.height = height
16
17     def area(self):
18         return self.width * self.height
19
20     def perimeter(self):
21         return 2 * (self.width + self.height)
22
23 class Circle(Shape):
24     def __init__(self, radius):
25         self.radius = radius
26
27     def area(self):
28         return 3.14 * self.radius ** 2
29
30     def perimeter(self):
31         return 2 * 3.14 * self.radius
32
...Program finished with exit code 0
Press ENTER to exit console.
```

Pada contoh di atas, kelas Shape merupakan sebuah kelas abstrak yang mewakili sebuah interface, yang memiliki dua metode abstrak, yaitu area dan perimeter. Kelas Rectangle dan Circle merupakan kelas turunan yang mengimplementasikan kelas abstrak Shape, sehingga wajib mengimplementasikan kedua metode abstrak tersebut. Kelas Rectangle dan Circle masing-masing mengimplementasikan metode area dan perimeter sesuai dengan kebutuhan objek yang mereka wakili, dan memastikan bahwa keduanya memenuhi kontrak yang telah didefinisikan oleh kelas Shape. Dalam contoh tersebut, penggunaan kelas abstrak Shape sebagai interface memastikan bahwa kelas-kelas turunan yang menerapkannya memiliki perilaku yang konsisten dan memenuhi kontrak yang telah ditentukan.

C. Metaclass

Metaclass adalah kelas yang mendefinisikan perilaku sebuah kelas, sedangkan kelas mendefinisikan perilaku sebuah objek. Metaclass biasanya digunakan sebagai pabrik kelas, sehingga memungkinkan untuk melakukan hal-hal ekstra saat membuat kelas seperti mendaftarkan kelas baru dengan beberapa registri atau mengganti kelas dengan sesuatu yang lain sama sekali. Dalam Python, metaclass dibuat dengan membuat subclass dari kelas `type`. Ketika pernyataan `class` dieksekusi, Python pertama mengeksekusi tubuh pernyataan `class` sebagai blok kode yang normal, kemudian metaclass dipanggil dengan nama, pangkalan, dan atribut dari kelas untuk melakukan instansiasi. Metaclass juga memungkinkan untuk mendefinisikan metode normal dan metode 'sihir' normal pada metaclass itu sendiri.