

MODUL MATA KULIAH

BAHASA PEMROGRAMAN DASAR

PG168 – 3 SKS



**FAKULTAS TEKNOLOGI INFORMASI
UNIVERSITAS BUDI LUHUR**

**JAKARTA
SEPTEMBER 2021**

TIM Penyusun

Agus Umar Hamdani, M.Kom



MODUL PERKULIAHAN #11

PEMROGRAMAN BERORIENTASI OBYEK DENGAN PYTHON

Capaian Pembelajaran	:	Mahasiswa Mampu: <ol style="list-style-type: none">1. Memahami konsep dasar pemrograman Object-Oriented dalam Python.2. Memahami karakteristik pemrograman berorientasi obyek yang ada pada bahasa pemrograman Python
Sub Pokok Bahasan	:	<ol style="list-style-type: none">1. <i>Class</i>2. <i>Penggunaan Class</i>
Daftar Pustaka	:	<ol style="list-style-type: none">1. Zarman, Wendi dan Wicaksono, Mochamad Fajar. "<i>Implementasi Algoritma dalam bahasa Python</i>". Edisi Pertama. Bandung : Penerbit Informatika. 2020.2. Kurniawati, Arik. "<i>Algoritma dan Pemrograman menggunakan Python</i>". Edisi Pertama. Yogyakarta : Depublish. 2016.3. Ismah. "<i>Pemrograman Komputer Dasar-dasar Python</i>". Jakarta : Fakultas Ilmu Pendidikan Universitas Muhammadiyah Jakarta. 2010.4. Irfani, M. Haviz dan Dafid. "<i>Modul Praktikum Dasar Pemrograman dengan bahasa Python</i>". Palembang : Sekolah Tinggi Manajemen Informatika Global Informatika Multidata Palembang. 2016.5. Fikri, Rijalul. "<i>Praktikum Algoritma dan Pemrograman Komputer</i>". Surabaya : Program Studi Teknik Komputer dan Telematika Jurusan Teknik Elektro Fakultas Teknologi Industri Institut Teknologi Sepuluh Nopember. 2010.6. Nuraini, Rini. 2010. Desk Check Table Pada Flowchart Operasi Perkalian Matriks. Jurnal Petir. Vol. 10(1). Sekolah Tinggi Teknik – PLN (STT-PLN).7. Romzi, Muhammad dan Kurniawan, Budi. 2020. Pembelajaran Pemrograman Python Dengan Pendekatan Logika Algoritma. JTIM : Jurnal Teknik Informatika Mahakarya. Vol. 03(2). Hal. 37-44.8. Programiz.com. Python Operators (https://www.programiz.com/python-programming/operators diakses pada 29 September 2021 pukul 21.32 WIB)9. Linux.die.net. Dive Into Python : Python from novice to pro. (https://linux.die.net/diveintopython/html/getting_to_know_python/everything_is_a_n_object diakses pada 19 Desember 2021).

PRAKTIKUM 10

PEMROGRAMAN BERORIENTASI OBYEK DENGAN PYTHON

12.1 Teori Singkat

Pemrograman berorientasi objek, atau *Object-Oriented Programming* (OOP) adalah paradigma pemrograman yang menyediakan sarana untuk menyusun program sehingga sifat (*properties, attributes*) dan perilaku (*method, operation*) digabungkan menjadi objek. sifat (*properties, attributes*) adalah identitas atau variabel dari suatu obyek. perilaku (*method, operation*) adalah kemampuan atau fitur yang dimiliki oleh obyek tersebut. Secara lebih sederhana, kita dapat definisikan bahwa obyek (object) adalah sebuah gabungan dari kumpulan variabel (dinamakan atribut) dan kumpulan operasi / method (dinamakan perilaku) [8]. Atas dasar definisi tersebut, maka dapat dikatakan bahwa semua hal di dalam bahasa Python adalah sebuah obyek [9]. Bahkan jika diperhatikan, setiap fungsi (Function / Procedure) pada bahasa Python memiliki atribut `__doc__` yang merupakan ciri dari sebuah obyek. Tipe Data juga merupakan obyek dari sebuah class dengan nama function. Berikut ini adalah terminologi atau istilah-istilah dalam Object-Oriented Programming (OOP) .

Tabel 12.1 Istilah-istilah dalam Object-Oriented Programming

Istilah	Deskripsi
<i>Class</i>	Cetak biru atau prototipe dari objek dimana kita mendefinisikan atribut dari suatu objek. Atribut ini terdiri dari data member (variabel) dan fungsi (metode).
<i>Class Variable</i>	Variabel yang dishare atau dibagi oleh semua instance (turunan) dari kelas. Variabel kelas didefinisikan di dalam kelas, tapi di luar metode-metode yang ada dalam kelas tersebut.
<i>Data member</i>	Variabel yang menyimpan data yang berhubungan dengan kelas dan objeknya.

Istilah	Deskripsi
<i>Function overloading</i>	Fungsi yang memiliki nama yang sama di dalam kelas, tapi dengan jumlah dan tipe argumen yang berbeda sehingga dapat melakukan beberapa hal yang berbeda.
<i>Operator overloading</i>	Beberapa fungsi atau kegunaan untuk suatu operator. Misalnya operator + dibuat tidak hanya untuk penjumlahan, tapi juga untuk fungsi lain.
<i>Instance variable</i>	Variabel yang didefinisikan di dalam suatu metode dan hanya menjadi milik dari instance kelas.
<i>Inheritance</i>	Pewarisan karakteristik sebuah kelas ke kelas lain yang menjadi turunannya.
<i>Instance</i>	Istilah lain dari objek suatu kelas. Sebuah objek yang dibuat dari prototipe kelas Lingkaran misalnya disebut sebagai instance dari kelas tersebut.
<i>Instantiation</i>	Pembuatan instance/objek dari suatu kelas.
<i>Method</i>	Jenis fungsi khusus (procedure atau function) yang didefinisikan dalam suatu kelas.
<i>Object</i>	instansiasi atau perwujudan dari sebuah kelas. Bila kelas adalah prototipenya, dan objek adalah barang jadinya.

12.2 PEMBUATAN KELAS

Kelas (Class) pada bahasa Python dapat dikatakan sebagai cetak biru (blueprint) dari object (atau instance) yang ingin dibuat. Kelas adalah cetakannya atau definisinya, sedangkan obyek (atau instance) adalah obyek nyatanya. Atribut merepresentasikan variabel yang dimiliki oleh sebuah obyek. Sedangkan perilaku merepresentasikan fungsi yang dimiliki sebuah obyek.

Simbol sebuah kelas dapat dilihat pada gambar 12.1.

Nama Class
Attribute / Properties
Method / Operation

Gambar 12.1 Simbol Kelas

Misalnya, kelas pegawai yang memiliki atribut / properti berupa : nama, jenis kelamin (jenkel), alamat, telepon dan gaji. Dengan perilaku pada kelas pegawai berupa prosedur atau fungsi yang berupa : `tampilkan_jumlah()` dan `tampilkan_profil()`, maka kita dapat menggambarkan model kelas pegawai sebagai berikut :

Class Name : Pegawai
Nama Jenkel Alamat Telepon Gaji
<code>tampil_jumlah()</code> <code>tampil_profil()</code>

Gambar 12.2 Contoh Kelas

Adapun perintah dalam bahasa Python yang digunakan untuk membuat / mendefinisikan kelas adalah :

```

Class classname :
    Def __init__ (self, parameter) :
        Isi yang ingin dimasukan
    Def method1 (self) :
        Isi method 1
    Def method2 (self) :
        Isi method 2
    Class_suite

```

Keterangan :

- Def method1(self) dan def method2(self) merupakan definisi fungsi atau prosedur yang akan diterapkan di kelas.
- *Class_Suite* terdiri dari semua pernyataan komponen yang mendefinisikan *class member*, *data attributes* dan fungsi.

Studi Kasus 12.1

Gambar 12.3 merupakan contoh pembuatan kelas pegawai berdasarkan definisi yang telah dilakukan pada gambar 12.2 menggunakan bahasa Python dengan nama module “Pegawai_class.py”.



```
1 class Pegawai:
2     #berisi properti umum untuk kelas pegawai
3     jumlah_pgw = 0
4
5     def __init__(self, nama, jenkel, alamat, telepon, gaji):
6         self.nama = nama
7         self.jenkel = jenkel
8         self.alamat = alamat
9         self.telepon = telepon
10        self.gaji = gaji
11        Pegawai.jumlah_pgw += 1
12
13    def tampil_jumlah(self):
14        print ("Total Pegawai %d" % Pegawai.jumlah_pgw)
15
16    def Tampil_Profil(self):
17        print ("Nama : ", self.nama, ", Jenis Kelamin: ", self.jenkel, "Alamat : ", self.alamat,
18              "Telepon : ", self.telepon, "Gaji : ", self.gaji)
19
20    pgw1 = Pegawai("Restu Singgih", "L", "Jakarta", "087809299090", 4500000)
21    pgw2 = Pegawai("Nilam Cahya", "P", "Cirebon", "081290902323", 7850000)
22
23    pgw1.Tampil_Profil()
24    pgw2.Tampil_Profil()
25    print("Total Pegawai : ", Pegawai.jumlah_pgw)
26
```

Gambar 12.3 Contoh Pembuatan Kelas Pegawai

Perhatikan gambar 12.3, Variabel jumlah_pgw adalah variabel kelas yang dibagikan ke semua instance/objek dari kelas ini. Variabel ini bisa diakses dari dalam atau luar kelas dengan menggunakan notasi titik, **Pegawai.jumlah_pgw**.

Metode `__init__()` adalah metode konstruktor, yaitu metode khusus yang digunakan Python untuk menginisialisasi pembuatan objek dari kelas tersebut. Fungsi – fungsi di dalam kelas (disebut *method, operation*) dapat didefinisikan sama seperti mendefinisikan fungsi pada umumnya. Hanya saja, harus ada argumen pertama bernama ***self***. Pada saat pemanggilan fungsi, argumen ini otomatis ditambahkan oleh Python. Kita tidak perlu menambahkannya pada saat memanggil fungsi.

```
22
23 #Membuat obyek untuk memanggil kelas
24 pgw1 = Pegawai("Restu Singgih", "L", "Jakarta", "087809299090", 4500000)
25 pgw2 = Pegawai("Nilam Cahya", "P", "Cirebon", "081290902323", 7850000)
26
```

Gambar 12.4 Contoh Pembuatan Kelas Pegawai

Perhatikan gambar 12.4, variabel `pgw1` dan `pgw2` merupakan contoh pembuatan (inisiasi) obyek yang memanggil kelas pegawai. Untuk membuat obyek dari sebuah kelas, maka kita bisa memanggil nama kelas dengan argument sesuai dengan fungsi `__init__` pada saat kita mendefinisikan kelas tersebut.

```
26
27 #Mengakses atribut yang ada pada obyek
28 pgw1.Tampil_Profil()
29 pgw2.Tampil_Profil()
30 print("Total Pegawai : ", Pegawai.jumlah_pgw)
31
```

Gambar 12.5 Contoh Pembuatan Kelas Pegawai

Perhatikan gambar 12.5, perintah `pgw1.Tampil_Profil()`, `pgw2.Tampil_Profil()` dan `Pegawai.jumlah_pgw` merupakan contoh mengakses atribut yang ada pada obyek kelas. Adapun hasil keluaran dari program diatas ditampilkan dapat dilihat pada gambar 12.6.

```
Nama : Restu Singgih , Jenis Kelamin: L Alamat : Jakarta Telepon : 087809299090 Gaji : 4500000
Nama : Nilam Cahya , Jenis Kelamin: P Alamat : Cirebon Telepon : 081290902323 Gaji : 7850000
Total Pegawai : 2
```

Gambar 12.6 Hasil Keluaran Program

Kita pun dapat menambah, mengubah dan menghapus atribut yang ada pada sebuah obyek, seperti perintah-perintah yang terlihat pada gambar 12.7.

```

31
32     #Mengubah atribut obyek
33     pgw1.nama = "Malin Kundang"
34     pgw1.gaji = 8550000
35     pgw1.Tampil_Profil()
36     #Menghapus obyek
37     del pgw2
38     pgw2.Tampil_Profil()
39

```

Gambar 12.7 Modifikasi Atribut Pada Obyek

Perhatikan gambar 12.7, kita ingin mengubah nilai atribut nama dan gaji pada obyek pgw1 dengan nilai baru dan kemudian menampilkan kembali profil dari obyek pgw1 tersebut. Sedangkan perintah **del pgw2** digunakan untuk menghapus obyek pgw2. Adapun hasil keluaran dari program diatas ditampilkan dapat dilihat pada gambar 12.8.

```

Traceback (most recent call last):
  File "C:\Users\Agus Umar\PycharmProjects\pythonProject12\Pegawai_class.py", line 38, in <module>
    pgw2.Tampil_Profil()
NameError: name 'pgw2' is not defined
Nama : Restu Singgih , Jenis Kelamin: L , Alamat : Jakarta Telepon : 087809299090 Gaji : 4500000
Nama : Nilam Cahya , Jenis Kelamin: P , Alamat : Cirebon Telepon : 081290902323 Gaji : 7850000
Total Pegawai : 2
Nama : Malin Kundang , Jenis Kelamin: L , Alamat : Jakarta Telepon : 087809299090 Gaji : 8550000

```

Gambar 12.8 Hasil Keluaran Program

Untuk melakukan modifikasi atribut pada obyek, maka kita dapat menggunakan fungsi-fungsi berikut :

Nama Fungsi	Deskripsi
getattr(obj, name[, default])	Mengakses atribut objek
hasattr(obj, name)	Memeriksa apakah objek memiliki atribut tertentu atau tidak

setattr(obj, name, value)	Mengatur nilai atribut. Jika atribut tidak ada, maka atribut tersebut akan dibuatkan.
delattr(obj, name)	Menghapus atribut dari objek

Setiap kelas di bahasa Python memiliki atribut *built-in* (bawaan) yang dapat diakses menggunakan operator titik. Berikut ini adalah atribut-atribut tersebut :

Tabel 12.2 Atribut Kelas *Built-In*

Nama Atribut <i>Built-In</i>	Deskripsi
<code>__dict__</code>	Dictionary yang berisi namespace dari kelas.
<code>__doc__</code>	Mengakses string dokumentasi (docstring) dari kelas.
<code>__name__</code>	Nama kelas.
<code>__module__</code>	Nama modul tempat kelas didefinisikan. Nilai atribut ini di mode interaktif adalah <code>"__main__"</code> .
<code>__bases__</code>	Dasar dari kelas, bila kelas tidak merupakan turunan dari kelas lain, maka induknya adalah kelas object.

Gambar 12.9 merupakan contoh penggunaan fungsi-fungsi built-in pada kelas pegawai.

```

39
40     #Menggunakan fungsi-fungsi built-in pada kelas
41     print("Pegawai.__doc__:", Pegawai.__doc__)
42     print("Pegawai.__name__:", Pegawai.__name__)
43     print("Pegawai.__module__:", Pegawai.__module__)
44     print("Pegawai.__dict__:", Pegawai.__dict__)
45     print("Pegawai.__bases__:", Pegawai.__bases__)
46

```

Gambar 12.9 Contoh Penggunaan Fungsi-fungsi Built-in pada Kelas

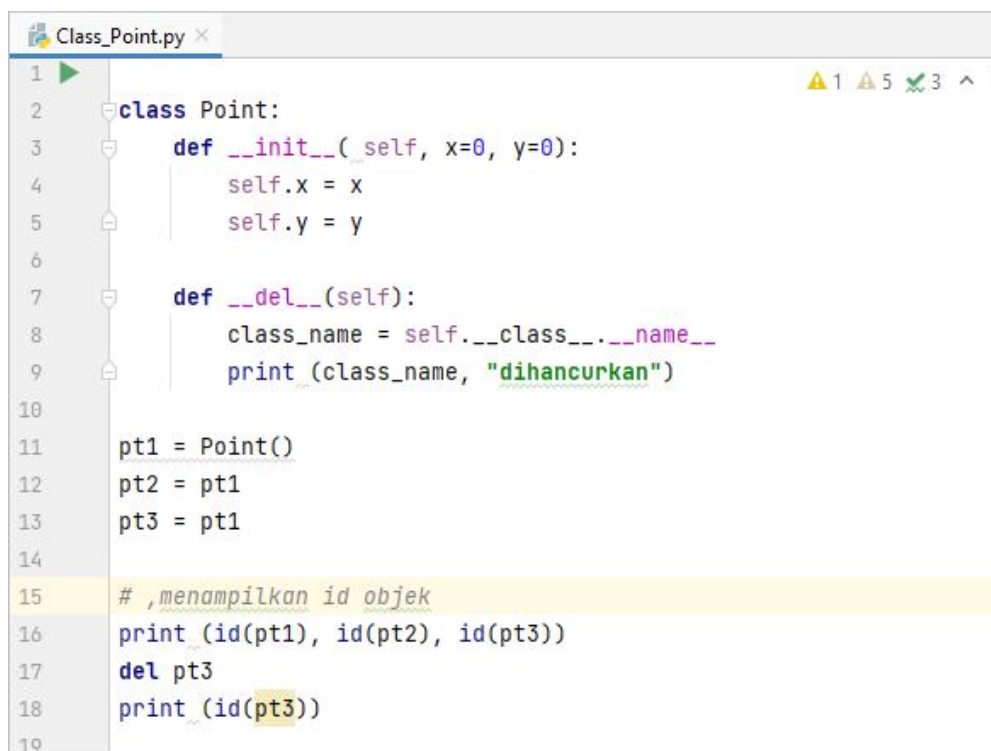
Adapun hasil keluaran dari program diatas ditampilkan dapat dilihat pada gambar 12.10.

```
Nama : Restu Singgih , Jenis Kelamin: L , Alamat : Jakarta Telepon : 087809299090 Gaji : 4500000
Nama : Nilam Cahya , Jenis Kelamin: P , Alamat : Cirebon Telepon : 081290902323 Gaji : 7850000
Total Pegawai : 2
Nama : Malin Kundang , Jenis Kelamin: L , Alamat : Jakarta Telepon : 087809299090 Gaji : 8550000
Pegawai.__doc__: Dasar kelas untuk semua karyawan
Pegawai.__name__: Pegawai
Pegawai.__module__: __main__
Pegawai.__dict__: {'__module__': '__main__', '__doc__': 'Dasar kelas untuk semua karyawan', 'jumlah_pgw': 2, '__init_
Pegawai.__bases__: (<class 'object'>,)
```

Gambar 12.10 Hasil Keluaran Program

Studi Kasus 12.2

Gambar 12.11 merupakan contoh penghapusan obyek dengan nama module “Point.py”.



```
1 class Point:
2     def __init__(self, x=0, y=0):
3         self.x = x
4         self.y = y
5
6     def __del__(self):
7         class_name = self.__class__.__name__
8         print(class_name, "dihancurkan")
9
10 pt1 = Point()
11 pt2 = pt1
12 pt3 = pt1
13
14 # ,menampilkan id objek
15 print(id(pt1), id(pt2), id(pt3))
16 del pt3
17 print(id(pt3))
18
19
```

Gambar 12.11 Contoh Penggunaan Destructor del

Python menghapus objek yang sudah tidak terpakai secara otomatis untuk menghemat memori. Proses ini disebut dengan pengumpulan sampah (*garbage collection*). Kolektor sampah Python terus berjalan pada saat program dieksekusi dan

dipicu pada saat tidak ada lagi referensi/variabel yang merujuk ke objek. Jumlah referensi terhadap objek bertambah pada saat ada variabel yang merujuk ke objek tersebut. Sebaliknya referensi terhadap objek berkurang ketika variabel terhapus dengan menggunakan `__del__()`, atau saat terjadi penugasan ulang, atau saat referensi keluar dari ruang lingkup-nya. Pada saat referensi terhadap objek sudah nol, maka Python akan otomatis menghapus objek tersebut.

Adapun hasil keluaran dari program diatas ditampilkan dapat dilihat pada gambar 12.12.

```
Traceback (most recent call last):
  File "C:\Users\Agus Umar\PycharmProjects\pythonProject12\Class_Point.py", line 18, in <module>
    print (id(pt3))
NameError: name 'pt3' is not defined
2201155850192 2201155850192 2201155850192
Point dihancurkan
```

Gambar 12.12 Hasil Keluaran Program

12.3 PEWARISAN (*INHERITANCE*)

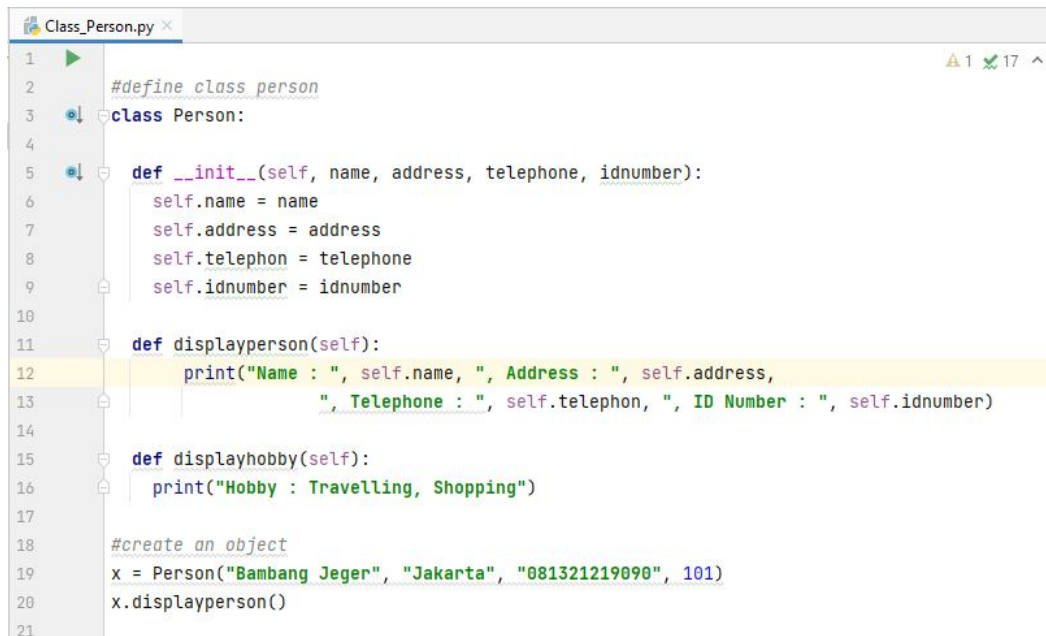
Dengan Pewarisan (Inheritance), kita dapat mendefinisikan kelas yang mewarisi semua metode dan properti dari kelas lain, tanpa harus membuat kelas baru dari awal. Turunannya disebut kelas anak (*child class*) dan yang mewariskannya disebut kelas induk (*parent class*). Kelas anak mewarisi atribut dari kelas induk, dan kita bisa menggunakan atribut tersebut seolah atribut itu didefinisikan juga di dalam kelas anak. Kelas anak juga bisa menimpa (*override*) data dan metode dari induknya dengan data dan metodenya sendiri. Satu kelas anak bisa mewarisi karakteristik dari satu atau beberapa kelas induk.

Pewarisan memiliki sintak penulisan sebagai berikut :

```
class SubClassName (ParentClass1[, ParentClass2, ...]):
    def method() :
        class_suite
```

Studi Kasus 12.3

Gambar 12.13 merupakan contoh pembuatan kelas **Person** yang berisi informasi person menggunakan bahasa Python yang disimpan ke dalam nama module “Class_Person.py”.



```
1 #define class person
2
3 class Person:
4
5     def __init__(self, name, address, telephone, idnumber):
6         self.name = name
7         self.address = address
8         self.telephon = telephone
9         self.idnumber = idnumber
10
11     def displayperson(self):
12         print("Name : ", self.name, ", Address : ", self.address,
13             ", Telephone : ", self.telephon, ", ID Number : ", self.idnumber)
14
15     def displayhobby(self):
16         print("Hobby : Travelling, Shopping")
17
18 #create an object
19 x = Person("Bambang Jeger", "Jakarta", "081321219090", 101)
20 x.displayperson()
21
```

Gambar 12.13 Contoh Pembuatan Kelas Person

Perhatikan gambar 12.13, terdapat deklarasi kelas dengan nama **Person** yang memiliki inisiasi berupa **__init__** yang disertai dengan atribut *name*, *address* dan *telephone* dan *ID Number*. Parameter **self** mewakili obyek atau instance dari kelas yang digunakan untuk mengakses variabel-variabel dan method / operasi milik kelas **Person**. Parameter **self** ini **wajib** dideklarasikan secara eksplisit. Terdapat deklarasi fungsi `displayperson()` untuk menampilkan nilai dari setiap atribut yang ada pada kelas dan fungsi `displayhobby()` untuk menampilkan informasi hobi. Variabel `x` merupakan obyek atau instance dari kelas `Person` yang berisi nama : “Bambang Jeger”, Address = “Jakarta”, Telepon = “081321219090” dan ID Number = “101”



```

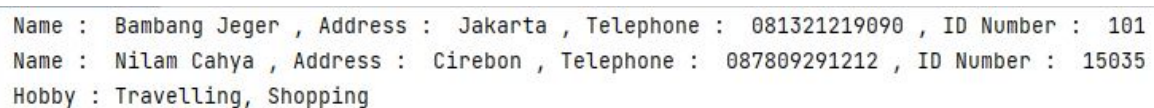
22
23     #Mendefinisikan subkelas
24     class Student(Person):
25
26         #definisi atribut subkelas
27         def __init__(self, name, address, telephone, idnumber, year):
28             #memanggil atribut superkelas
29             super().__init__(name, address, telephone, idnumber)
30             self.address = address
31             self.graduationyear = year
32
33     y = Student("Nilam Cahya", "Cirebon", "087809291212", 15035, 2019)
34     y.displayperson()
35     y.displayhobby()
36

```

Gambar 12.14 Contoh Pembuatan Kelas Student

Perhatikan gambar 12.14, terdapat deklarasi kelas dengan nama **Student** yang memiliki inisiasi berupa **__init__** yang disertai dengan atribut *name*, *address*, *telephone*, *idnumber* dan *year* dan disertai dengan parameter kelas **Person**. Parameter **self** mewakili obyek atau instance dari kelas yang digunakan untuk mengakses variabel-variabel dan method / operasi yang ada di kelas **Student**. Fungsi **super()** digunakan untuk mengacu ke kelas induk dari suatu objek. Fungsi ini akan mengembalikan objek superclass yang memungkinkan kita untuk mengakses property dan method / operasi yang ada kelas **Person**. Kedua kode program diatas masih disimpan dalam satu module yaitu **Class_Person.py**.

Adapun hasil keluaran dari program diatas ditampilkan dapat dilihat pada gambar 12.15.



```

Name : Bambang Jeger , Address : Jakarta , Telephone : 081321219090 , ID Number : 101
Name : Nilam Cahya , Address : Cirebon , Telephone : 087809291212 , ID Number : 15035
Hobby : Travelling, Shopping

```

Gambar 12.15 Hasil Keluaran Program

Dengan cara yang sama, kita dapat mengarahkan sebuah kelas dari beberapa kelas Induk sebagai berikut :

```

class A:          # define your class A
.....

```

```
class B:          # define your class B
.....

class C(A, B):    # subclass of A and B
.....
```

Gambar 12.16 Penggunaan Class dan Subclass

Anda dapat menggunakan fungsi `issubclass()` atau `isinstance()` untuk memeriksa hubungan dua kelas dan instance :

- Fungsi boolean **`issubclass(sub, sup)`** mengembalikan nilai True, jika sub subclass yang diberikan memang merupakan subclass dari superclass sup.
- Fungsi boolean **`isinstance(obj, Class)`** mengembalikan nilai True, jika obj adalah turunan dari kelas atau merupakan turunan dari subkelas class.

12.4 OVERRIDING METHOD

Kita selalu dapat mengganti method / operasi yang ada di kelas induk. Salah satu alasan untuk mengganti method atau operasi di kelas induk adalah karena kita mungkin menginginkan fungsionalitas khusus atau berbeda di subkelas. *Overriding* adalah properti kelas untuk mengubah implementasi metode yang disediakan oleh salah satu kelasnya. Dengan menggunakan *Overriding Method*, kelas dapat "menyalin" kelas lain, menghindari kode duplikat, dan pada saat yang sama meningkatkan atau menyesuaikan bagian dari itu. *Overriding Method* merupakan bagian dari mekanisme pewarisan. Dalam metode Python, penggantian terjadi hanya dengan mendefinisikan di kelas anak sebuah metode dengan nama metode yang sama di kelas induk.

Studi Kasus 12.4

Gambar 12.17 merupakan contoh penggunaan *Overriding Method* pada module "Overriding_Method.py".

```
1 #define Parent class
2 class Parent(object):
3     def __init__(self):
4         self.value = 4
5     def get_value(self):
6         return self.value
7
8 parent1 = Parent()
9 print("Parent : ", parent1.get_value())
10
11 #define Child class
12 class Child(Parent):
13     def get_value(self):
14         return self.value + 1
15
16 child1 = Child()
17 print("Child : ", child1.get_value())
18
```

Gambar 12.17 Contoh Penggunaan *Overriding Method*

Perhatikan gambar 12.17, terdapat deklarasi kelas dengan nama **Parent** dengan parameter **object** dan terdapat method / operasi yang terdiri **__init__()** dengan parameter **self** dan **get_value()** dengan parameter **self** yang mengembalikan nilai dari value yang diberikan. Pada deklarasi kelas kedua diberi nama **Child** dengan parameter **Parent** dan terdapat method / operasi dengan nama **get_value()**. Terdapat penggunaan method / operasi dengan nama yang sama, yaitu **get_value()**, namun keduanya memiliki tugas yang berbeda. Adapun hasil keluaran dari program diatas ditampilkan dapat dilihat pada gambar 12.15.

```
Value for Parent : 4
Value for Child : 5
```

Gambar 12.15 Hasil Keluaran Program

Tabel berikut mencantumkan beberapa fungsionalitas umum yang dapat kita gunakan untuk keperluan Overloading Method.

Tabel 12.3 Daftar Fungsi Untuk Keperluan Overloading Method

Nama Method	Deskripsi	Contoh Penggunaan
<code>__init__(self [, args..])</code>	Konstruktor dengan beberapa argument / parameter.	<code>obj = classname(args)</code>
<code>__del__(self)</code>	Destruktor, menghapus sebguah obyek.	<code>del obj</code>
<code>__repr__(self)</code>	Mengevaluasi representasi string.	<code>repr(obj)</code>
<code>__str__(self)</code>	Mencetak repretasi string.	<code>str(obj)</code>
<code>__cmp__(self, x)</code>	Membandingkan dua buah obyek.	<code>Cmp(obj, x)</code>

12.5 OVERLOADING

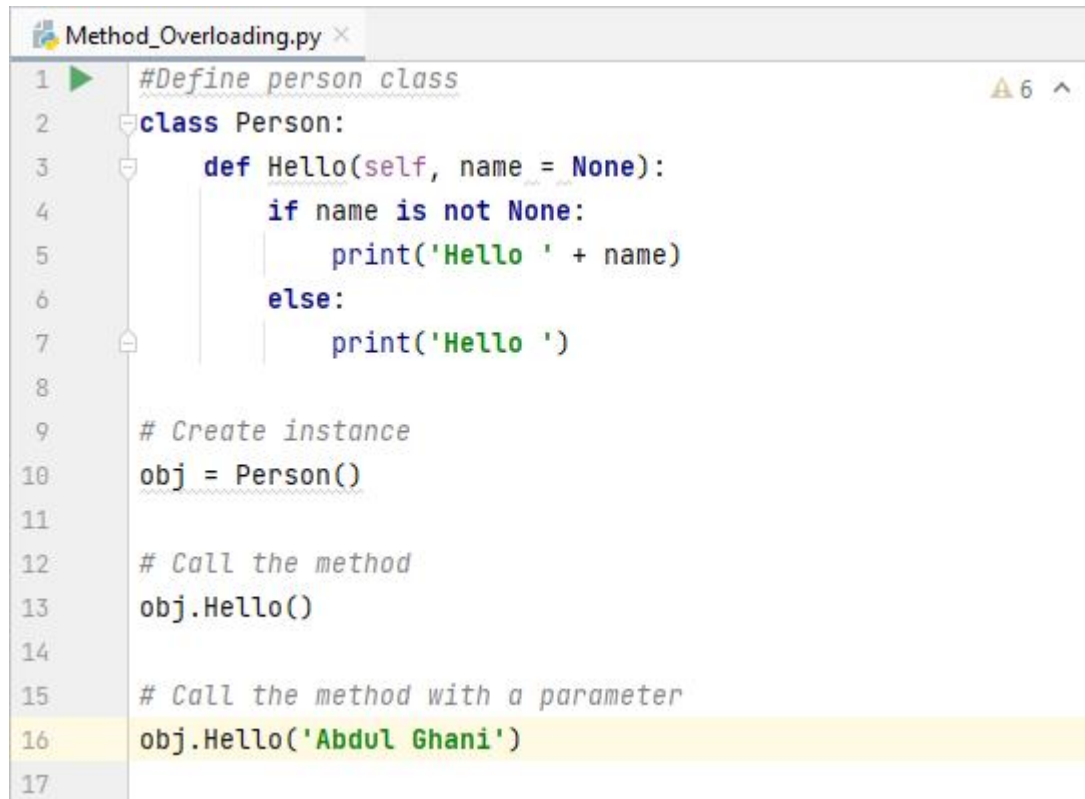
Overloading adalah kemampuan suatu fungsi atau operator untuk berperilaku dengan cara yang berbeda berdasarkan parameter yang diteruskan ke fungsi, atau operand yang dijalankan oleh operator.

Beberapa keuntungan menggunakan *overloading* adalah:

- Overloading* merupakan suatu metode yang dapat digunakan Kembali (reusable). Misalnya, daripada menulis beberapa metode yang hanya sedikit berbeda, kita dapat menulis satu metode dan menggunakannya.
- Overloading* juga meningkatkan kejelasan kode dan menghilangkan kerumitan.
- Overloading* adalah konsep yang sangat berguna. Namun, ia memiliki sejumlah kelemahan yang terkait dengannya.
- Overloading* dapat membuat kebingungan saat digunakan melintasi batas pewarisan. Ketika digunakan secara berlebihan, menjadi rumit untuk mengelola fungsi yang overload.

Studi Kasus 12.5

Gambar 12.17 merupakan contoh penggunaan *Method Overloading* pada module “**Method_Overloading.py**”.



```
1 #Define person class
2 class Person:
3     def Hello(self, name = None):
4         if name is not None:
5             print('Hello ' + name)
6         else:
7             print('Hello ')
8
9 # Create instance
10 obj = Person()
11
12 # Call the method
13 obj.Hello()
14
15 # Call the method with a parameter
16 obj.Hello('Abdul Ghani')
```

Gambar 12.17 Contoh Penggunaan *Method Overloading*

Perhatikan gambar 12.17, terdapat deklarasi kelas dengan nama **Person** yang memiliki method / operasi berupa **Hello()** dengan parameter **self** dan **name** dengan nilai defaultnya adalah none. Kemudian terdapat kontrol percabangan untuk memeriksa nilai dari parameter name. Adapun hasil keluaran dari program diatas ditampilkan dapat dilihat pada gambar 12.18.

```
Hello
Hello Abdul Ghani
```

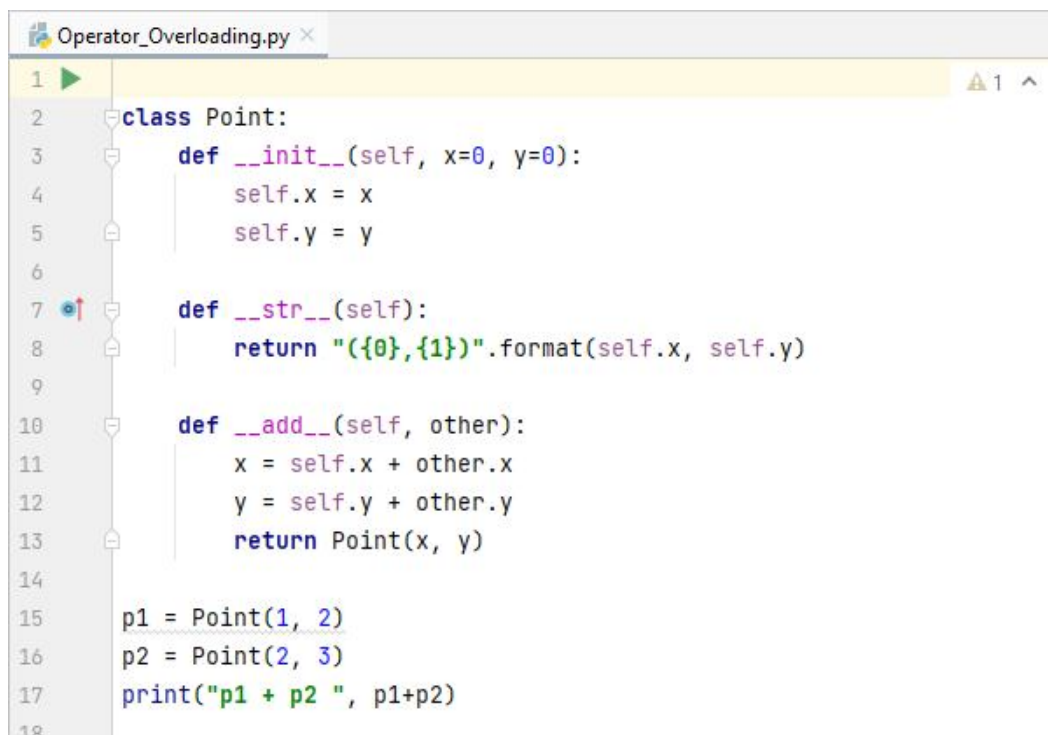
Gambar 12.18 Hasil Keluaran Program

Operator python berfungsi untuk kelas bawaan. Tetapi operator yang sama berperilaku berbeda dengan tipe yang berbeda. Misalnya, operator + akan melakukan penjumlahan aritmatika pada dua angka, menggabungkan dua daftar, atau menggabungkan dua string. Fitur dalam Python yang memungkinkan operator yang

sama memiliki arti yang berbeda sesuai dengan konteksnya disebut operator overloading. Kita bisa mendefinisikan metode `__add__` dalam kelas kita untuk melakukan penjumlahan vektor dan kemudian operator `+` akan berfungsi sesuai kehendak kita.

Studi Kasus 12.6

Gambar 12.19 merupakan contoh penggunaan *Operator Overloading* pada module “**Operator_Overloading.py**”.



```
1 class Point:
2     def __init__(self, x=0, y=0):
3         self.x = x
4         self.y = y
5
6     def __str__(self):
7         return "({0},{1})".format(self.x, self.y)
8
9     def __add__(self, other):
10        x = self.x + other.x
11        y = self.y + other.y
12        return Point(x, y)
13
14 p1 = Point(1, 2)
15 p2 = Point(2, 3)
16 print("p1 + p2 ", p1+p2)
```

Gambar 12.19 Contoh Penggunaan Operator Overloading

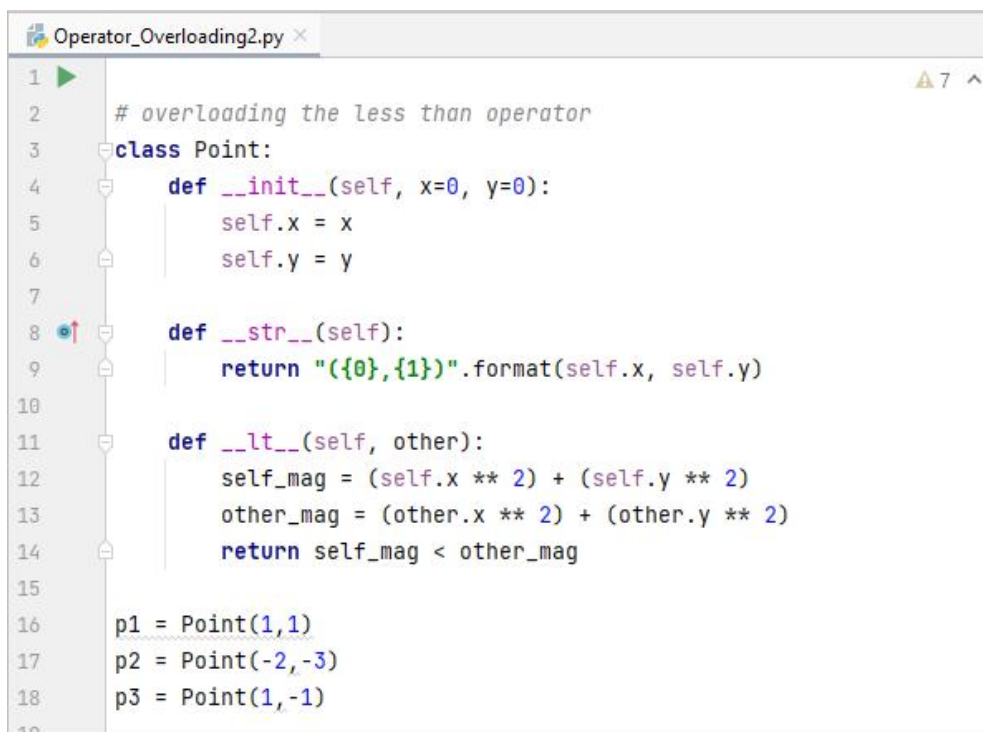
Perhatikan gambar 12.19, terdapat deklarasi kelas dengan nama **Point** yang memiliki method / operasi berupa `__init__()` dengan parameter **self** dan **x, y**, dimana nilai default untuk x dan y adalah 0. Atribut `self.x` diisi dengan nilai x dan atribut `self.y` diisi dengan nilai y. method / operasi `__str__()` dengan parameter `self` mengembalikan nilai `"({0},{1})".format(self.x, self.y)`. method / operasi `__add__()` dengan parameter **self** dan **other**, dimana variabel x diisi dengan nilai `self.x + other.x` dan variabel y diisi dengan `self.y + other.y` dan mengembalikan nilai `point(x, y)`. kemudian terdapat variabel obyek p1 dan p2. p1 mewakili kelas `Point()` dengan nilai parameter 1 dan 2, p2 mewakili kelas `Point()` dengan nilai parameter 2 dan 3. Kemudiannya hasil

penambahan dari kedua variabel obyek tersebut dicetak. Adapun hasil keluaran dari program diatas ditampilkan dapat dilihat pada gambar 12.20.

```
p1 + p2 : (3,5)
```

Gambar 12.20 Hasil Keluaran Program

Python tidak membatasi overloading operator ke operator aritmatika saja, kita juga dapat membebani operator perbandingan. Misalkan kita ingin mengimplementasikan simbol < di dalam kelas Point sebelumnya, maka kita bisa bandingkan besarnya titik-titik ini dari titik asal dan kembalikan hasilnya seperti terlihat pada gambar 12.21.



```
1 # overloading the less than operator
2 class Point:
3     def __init__(self, x=0, y=0):
4         self.x = x
5         self.y = y
6
7     def __str__(self):
8         return '({0},{1})'.format(self.x, self.y)
9
10    def __lt__(self, other):
11        self_mag = (self.x ** 2) + (self.y ** 2)
12        other_mag = (other.x ** 2) + (other.y ** 2)
13        return self_mag < other_mag
14
15    p1 = Point(1,1)
16    p2 = Point(-2,-3)
17    p3 = Point(1,-1)
18
19
```

Gambar 12.21 Contoh Penggunaan Operator <

Adapun hasil keluaran dari program diatas ditampilkan dapat dilihat pada gambar 12.22.

```
True
False
False
```

Gambar 12.22 Hasil Keluaran Program

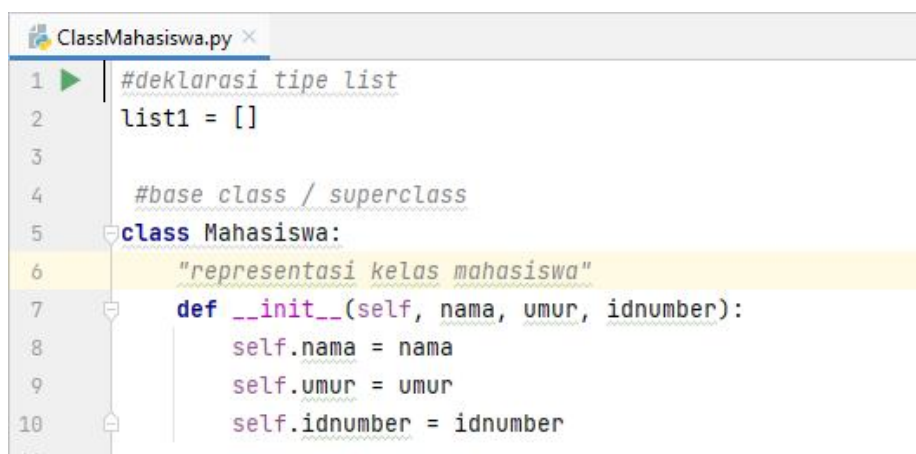
12.6 Praktikum

Langkah-langkah Praktikum

1. Buka Editor Python (IDLE / Pycharm / VSCode).
2. Buatlah file baru dengan membuka menu File > New > Source File atau dengan shortcut Ctrl + N.
3. Tulislah kode program berikut ini :

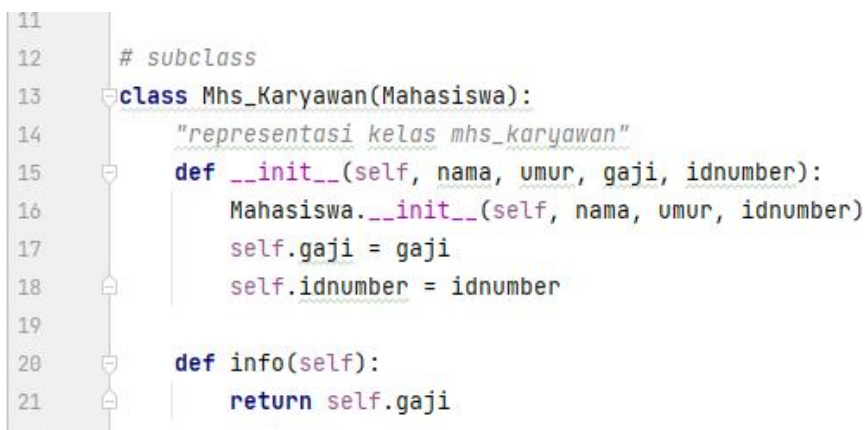
Program 12.1 : Praktikum12-1. Py

1. Buatlah modul program berisi kelas Mahasiswa dengan kode program sebagai berikut :



```
1  #deklarasi tipe list
2  list1 = []
3
4  #base class / superclass
5  class Mahasiswa:
6      "representasi kelas mahasiswa"
7      def __init__(self, nama, umur, idnumber):
8          self.nama = nama
9          self.umur = umur
10         self.idnumber = idnumber
11
```

Gambar 12.23 Membuat Kelas Mahasiswa



```
11
12 # subclass
13 class Mhs_Karyawan(Mahasiswa):
14     "representasi kelas mhs_karyawan"
15     def __init__(self, nama, umur, gaji, idnumber):
16         Mahasiswa.__init__(self, nama, umur, idnumber)
17         self.gaji = gaji
18         self.idnumber = idnumber
19
20     def info(self):
21         return self.gaji
22
```

Gambar 12.24 Membuat Kelas Mhs_Karyawan

```

23 # subclass
24 class Mhs_Pascasarjana(Mahasiswa):
25     "representasi kelas mhs_pascasarjana"
26     def __init__(self, nama, umur, hari, idnumber):
27         Mahasiswa.__init__(self, nama, umur, idnumber)
28         self.hari = hari
29         self.idnumber = idnumber
30
31     def info(self):
32         return self.hari
33

```

Gambar 12.25 Membuat Kelas Mhs_Pascasarjana

```

33
34 #membuat obyek mahasiswa
35 mhs1 = Mhs_Karyawan('Budi', 25, 3000000, '1711512020')
36 mhs2 = Mhs_Pascasarjana('Andi', 40, "Sabtu", '1761180801')
37 Mahasiswa = [mhs1, mhs2]
38 for orang in Mahasiswa :
39     print(orang.idnumber, orang.nama, orang.umur, orang.info())
40

```

Gambar 12.26 Membuat Kelas Mahasiswa

- Simpan kode program diatas dengan nama Praktikum12-1.py
- Jalankan program praktikum12-1 di atas, kemudian tuliskan apa yang tercetak di layar pada saat memanggil modul **"ClassMahasiswa.py"**

- Jelaskan kesimpulan anda.

12.7 Rangkuman

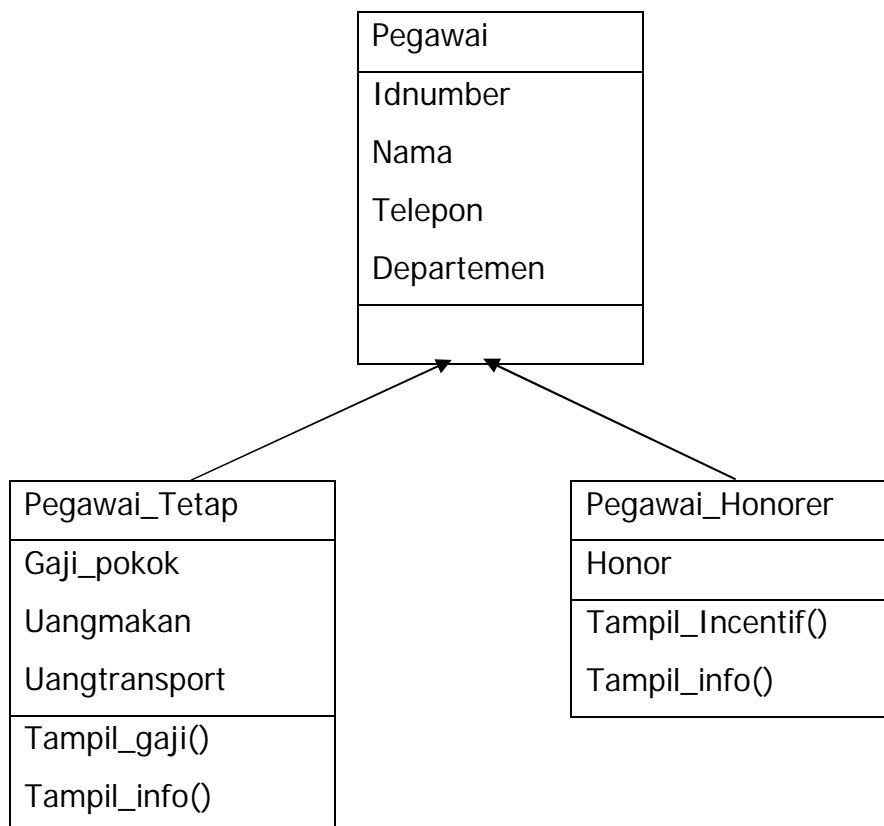
Adapun kesimpulan dari pertemuan 12 ini, antara lain :

- Objek pada python adalah kumpulan dari variabel-variabel (dinamakan atribut) dan kumpulan dari fungsi-fungsi (dinamakan perilaku).

- b. Atas definisi itu, maka semua hal di dalam python adalah sebuah Objek.
- c. Objek dan Kelas dalam Python bermakna sama. Akan tetapi, jika disebutkan dalam konteks terpisah, maka kelas adalah *blueprint* dan objek adalah variabel nyata.
- d. Konstruktor adalah fungsi yang pertama kali dipanggil ketika sebuah objek diinstantiasi.
- e. Objek bisa memiliki atribut yang berupa instan dari kelas lainnya.
- f. Kita dapat menggunakan sifat *inheritance*, *overriding method*, *method overloading* dan *operator overloading* untuk membuat kelas yang dinamis.

12.8 Latihan

1. Buatlah program untuk membuat kelas pegawai beserta subclass dengan struktur sebagai berikut :



Gambar 12.28 Contoh Struktur Kelas Pegawai

Keterangan :

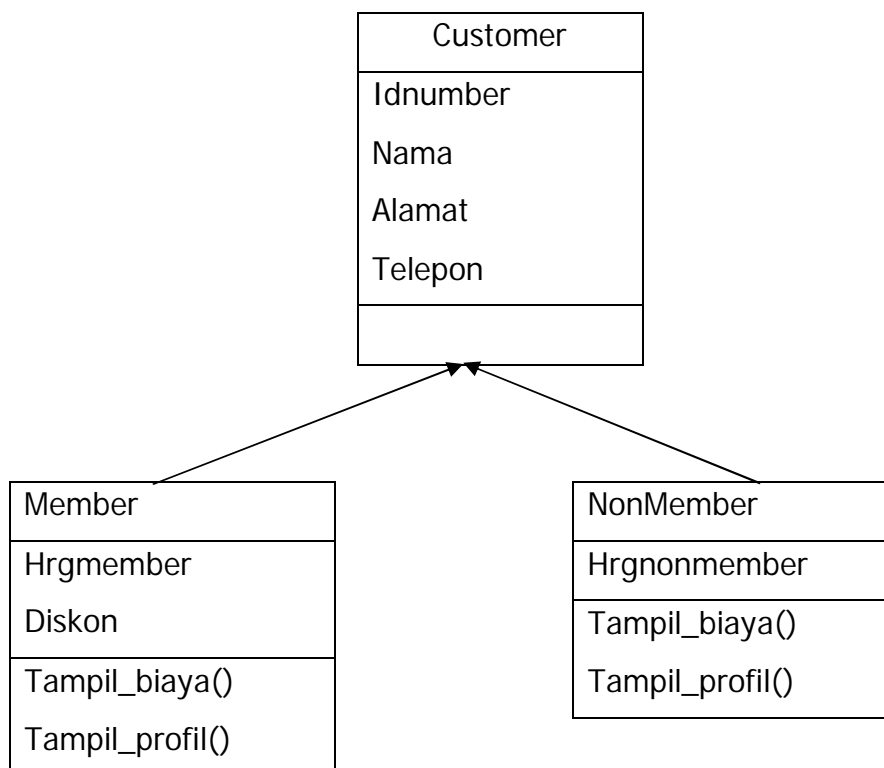
- Superclass : Pegawai
- Subclass : Pegawai_Tetap, Pegawai_Honorer
- Fungsi tampil_gaji() berisi jumlah gaji = gaji_pokok + uangmakan + uangtransport
- Fungsi tampil_incentif berisi jumlah incentive = honor
- Fungsi tampil_info berisi informasi mengenai identitas dari kelas pegawai.

Cetak informasi kelas pegawai tersebut menggunakan tipe List.

12.9 Tugas Mandiri

Kerjakan soal-soal berikut :

1. Buatlah modul program untuk membuat kelas Customer dengan struktur sebagai berikut :



Gambar 12.29 Contoh Struktur Kelas Member

Keterangan :

- Superclass : Customer
- Subclass : Member, NonMember
- Fungsi tampil_biaya() pada subclass member berisi total biaya =
hrgmember - diskon
- Fungsi tampil_biaya() pada subclass Nonmember berisi total biaya =
hrgnonmember
- Fungsi tampil_profil berisi informasi mengenai identitas dari kelas
customer.



FAKULTAS TEKNOLOGI INFORMASI
UNIVERSITAS BUDI LUHUR

Jl. Raya Ciledug, Petukangan Utara, Pesanggrahan
Jakarta Selatan, 12260
Telp: 021-5853753 Fax : 021-5853752
<http://fti.budiluhur.ac.id>