LAPORAN

Pemrograman Berorientasi Objek Praktik

Pertemuan Ke VI

Disusun Oleh :

5210411174_VERATINA FRIDAYANTI

PROGRAM STUDI INFORMATIKA
FAKULTAS SAINS & TEKNOLOGI
UNIVERSITAS TEKNOLOGI YOGYAKARTA

1. Teori
    - Polymorphims merupakan kemampuan suatu method untuk bekerja lebih dari suatu tipe argumen. Konsep terebut biasanya disebut overloading.
    - Polymorphims adalah suatu objek yang dapat memiliki berbagai bentu objek.
    - Overriding terjadi ketika deklarasi method subclass dengan nama dan parameter yang sama dengan method dari superclass
2. Tujuan
    - Agar program terlihat lebih rapi
    - Menghindari duplikat objek
3. Kode Program
    1. **implementasi kelas abstrak**

    - Source Code

```
from abc import ABC, abstractmethod
class Bentuk(ABC):
    @abstractmethod
    def luas(self):
        return self.__sisi * self.__sisi

    @abstractmethod
    def keliling(self):
        return 4 * self.__sisi

class Persegi(Bentuk):
    def __init__(self, sisi):
        self.__sisi = sisi
    def luas(self):
        return self.__sisi * self.__sisi
    def keliling(self):
        return 4 * self.__sisi

persegi = Persegi(6)
print(persegi.luas())
print(persegi.keliling())
```

    - **Penjelasan**

      Import ABC untuk mengimport storage yang dimiliki oleh python. Kemudian class bentuk mengambil data dari storage ABC, kemudian menampung method abstrak.

- **Output**

```
PS D:\KULIAH\Semester 2\Pemrograman Berorientasi Objek Praktik\Overloading&Overriding> & "C
indowsApps/python3.10.exe" "d:/KULIAH/Semester 2/Pemrograman Berorientasi Objek Praktik/Ove
y"
36
24
PS D:\KULIAH\Semester 2\Pemrograman Berorientasi Objek Praktik\Overloading&Overriding>
```

2. implementasi overloading class mahasiswa

- Source Code

```python
# Implementasi Overloading

class Mahasiswa:
    def __init__(self, nama, nim):
        self.nama = nama
        self.nim = nim

    def tampilMhs(self):
        print("Nama:", self.nama, ", nim:", self.nim)

# Method Overloading
    def hitungSKS(self, jmlsks=None, sks=None):
        if jmlsks !=None and sks!=None:
            totalsks = jmlsks + sks
            print("Total sks =", totalsks)
        else:
            totalsks = jmlsks
            print("Total sks =", totalsks)

        if totalsks >= 100:
            print("Diperbolehkan mengambil skripsi")

        else:
            print("Tidak diperbolehkan mengambil skripsi")

mhs1 = Mahasiswa("Eren", 123180015)
mhs2 = Mahasiswa("Luffy", 123190007)
mhs1.tampilMhs()
mhs2.tampilMhs()
mhs1.hitungSKS(80, 34) # Overloading
mhs2.hitungSKS(83)     # Overloading
```

- **Penjelasan**

  Class mahasiswa digunakan untuk menampung sebuah fungsi yang berisi tentang objek yang nantinya akan digunakan untuk menyimpan nama dan nim. Selanjutnya fungsi hitungSKS digunakan untuk menghitung total sls yang di dapatkan oleh mahasiswa. Dan juga digunakan untuk memberi info apakah mahasiswa ini bisa mengambil skripsi atau tidak.

- Output

```
PS D:\KULIAH\Semester 2\Pemrograman Berorientasi Objek Praktik\Overloading&Overriding> & "C:/Users/A
indowsApps/python3.10.exe" "d:/KULIAH/Semester 2/Pemrograman Berorientasi Objek Praktik/Overloading&
ss mahasiswa.py"
Nama: Eren , nim: 123180015
Nama: Luffy , nim: 123190007
Total sks = 114
Diperbolehkan mengambil skripsi
Total sks = 83
Tidak diperbolehkan mengambil skripsi
PS D:\KULIAH\Semester 2\Pemrograman Berorientasi Objek Praktik\Overloading&Overriding>
```

3. implementasi overloading class pegawai

- Source Code

```python
# Implementasi Overloading

class Pegawai:
    jumlah = 0

    def __init__(self, nama, gaji):
        self.nama = nama
        self.gaji = gaji
        Pegawai.jumlah += 1

    def tampilJumlah(self):
        print("Total pegawai", Pegawai.jumlah)

    def tampilpegawai(self):
        print("Nama: ", self.nama, ", gaji:", self.gaji)

    def tunjangan(self, istri=None, anak=None):
        if anak != None and istri != None:
            total = anak + istri
            print("Tunjangan anak + istri =", total)
        else:
            total = istri
            print("Tunjangan istri =", total)

# Memanggil kelas
peg1 = Pegawai("Eren", 2000)
peg2 = Pegawai("Luffy", 6000)
peg1.tampilpegawai()
peg2.tampilpegawai()
peg1.tunjangan(2500,2000)
peg2.tunjangan(2500)

print("Total pegawai %d" % Pegawai.jumlah)
rataGaji = (peg1.gaji + peg2.gaji)/Pegawai.jumlah
print("Rata-rata gaji ="+ str(rataGaji))
```

- Penjelasan

  Class mahasiswa digunakan untuk menampung sebuah fungsi yang berisi tentang objek yang nantinya akan digunakan untuk menyimpan nama dan nim. Selanjutnya fungsi hitungSKS digunakan untuk menghitung total sls yang di dapatkan oleh mahasiswa. Dan juga digunakan untuk memberi info apakah mahasiswa ini bisa mengambil skripsi atau tidak.

- Output

```
PS D:\KULIAH\Semester 2\Pemrograman Berorientasi Objek Praktik\Overloading&Overriding> &
indowsApps/python3.10.exe" "d:/KULIAH/Semester 2/Pemrograman Berorientasi Objek Praktik/
ss pegawai.py"
Nama:  Eren , gaji: 2000
Nama:  Luffy , gaji: 6000
Tunjangan anak + istri = 4500
Tunjangan istri = 2500
Total pegawai 2
Rata-rata gaji =4000.0
PS D:\KULIAH\Semester 2\Pemrograman Berorientasi Objek Praktik\Overloading&Overriding>
```

4. implementasi overriding class segiempat

- Source Code

```
class Segiempat():
    def __init__(self, panjang, lebar):
        self.panjang = panjang
        self.lebar   = lebar

    def hitungLuas(self):
        print("Luas Segiempat =", self.panjang * self.lebar, "m^2")

class Bujursangkar(Segiempat):
    def __init__(self, sisi):
        self.side = sisi
        Segiempat.__init__(self, sisi, sisi)

    def hitungLuas(self):
        print("Luas bujur sangkar =", self.side*self.side, "m^2")

b = Bujursangkar(4)
s = Segiempat(2,4)
b.hitungLuas()
s.hitungLuas()
```

- Penjelasan

  Class Segiempat digunakan untuk nemunpang fungsi yang nantinya akan digunakan untuk menghitung luas dari bangun ruang segiempat. Class bujursangkar menganmbil data dari class segiempat]

- Output

```
PS D:\KULIAH\Semester 2\Pemrograman Berorientasi Objek Praktik\Overloading&Overriding> & "C:/Us
indowsApps/python3.10.exe" "d:/KULIAH/Semester 2/Pemrograman Berorientasi Objek Praktik/Overloa
s segiempat.py"
Luas bujur sangkar = 16 m^2
Luas Segiempat = 8 m^2
PS D:\KULIAH\Semester 2\Pemrograman Berorientasi Objek Praktik\Overloading&Overriding>
```

5. overloadingComputerpart

- Souce Code

```
class ComputerPart():
    def __init__(self,nama,pabrikan,jenis,harga):
        self.pabrikan = pabrikan
        self.harga = harga
        self.nama = nama
        self.jenis = jenis

class Prosesor(ComputerPart):
    def __init__(self, nama, pabrikan, harga,speed,jumlah_core):
        super().__init__(nama, pabrikan,'processor', harga)
        self.jumlah_core = jumlah_core
        self.speed = speed
    def hitungharga(self, diskon=None ):
        jumlah_barang = int(input(f'masukan jumlah {self.nama} yang
di beli : '))
        hargatotal = jumlah_barang*self.harga
        if hargatotal > 500000:
            diskon =  hargatotal*0.4
            print('anda mendapatkan total diskon 40%')
            print('total harga :',diskon)
        else:
            diskon = diskon

class RandomAccessMemory(ComputerPart):
    def __init__(self, nama, pabrikan, harga,kapasitas):
        super().__init__(nama, pabrikan,'RAM', harga)
        self.kapasitas = kapasitas
    def hitungharga(self, diskon=None ):
        jumlah_barang = int(input(f'masukan jumlah {self.nama}  yang
di beli : '))
        hargatotal = jumlah_barang*self.harga
        if hargatotal > 500000:
            diskon =  hargatotal*0.4
            print('anda mendapatkan total diskon 40%')
            print('total harga :',diskon)
        else:
```

```
            diskon = diskon

class HardDiskSATA(ComputerPart):
    def __init__(self, nama,pabrikan, harga,kapasitas,rpm):
        super().__init__(nama, pabrikan,'SATA', harga)
        self.kapasitas = kapasitas
        self.rpm = rpm
    def hitungharga(self, diskon=None ):
        jumlah_barang = int(input(f'masukan jumlah {self.nama} yang
di beli : '))
        hargatotal = jumlah_barang*self.harga
        if hargatotal > 500000:
            diskon =  hargatotal*0.4
            print('anda mendapatkan total diskon 40%')
            print('total harga :',diskon)
        else:
            diskon = diskon

p = Prosesor('Intel','Core i 7 7740X',4350000,4,'4.3GHz')
m = RandomAccessMemory('V - Gen','DDR S0Dimm
PC19200/2400MHz',328000,'4GB')
hdd = HardDiskSATA('seagate','HDD 2.5 inc',295000,'500GB',7200)
parts = [p,m,hdd]
for i in parts:
    print('-'*25)
    print('{} {} produksi {} harga perbarang
{}'.format(i.jenis,i.nama,i.pabrikan,i.harga))
    i.hitungharga()
```

- Output

```
PS D:\KULIAH\Semester 2\Pemrograman Berorientasi Objek Praktik\Overloading&Overriding> & "C:/Users/ASUS
indowsApps/python3.10.exe" "d:/KULIAH/Semester 2/Pemrograman Berorientasi Objek Praktik/Overloading&Over
-------------------------
processor Intel produksi Core i 7 7740X harga perbarang 4350000
masukan jumlah Intel yang di beli : 5
anda mendapatkan total diskon 40%
total harga : 8700000.0
-------------------------
RAM V - Gen produksi DDR S0Dimm PC19200/2400MHz harga perbarang 328000
masukan jumlah V - Gen  yang di beli : 4
anda mendapatkan total diskon 40%
total harga : 524800.0
-------------------------
SATA seagate produksi HDD 2.5 inc harga perbarang 295000
masukan jumlah seagate yang di beli : 6
anda mendapatkan total diskon 40%
total harga : 708000.0
PS D:\KULIAH\Semester 2\Pemrograman Berorientasi Objek Praktik\Overloading&Overriding>
```

6. overridingComputerpart

- Source Code

```
class ComputerPart():
    def __init__(self,nama,pabrikan,jenis,harga):
```

```
        self.pabrikan = pabrikan
        self.harga = harga
        self.nama = nama
        self.jenis = jenis
        self.jumlah = int(input('masukan jumlah
{}:'.format(self.nama)))

    def hitungharga(self):
        print('harga total : ',self.harga*self.jumlah)

class Prosesor(ComputerPart):
    def __init__(self, nama, pabrikan, harga,speed,jumlah_core):
        super().__init__(nama, pabrikan,'processor', harga)
        self.jumlah_core = jumlah_core
        self.speed = speed
    def hitungharga(self):
        print('harga total : ',self.harga*self.jumlah)

class RandomAccessMemory(ComputerPart):
    def __init__(self, nama, pabrikan, harga,kapasitas):
        super().__init__(nama, pabrikan,'RAM', harga)
        self.kapasitas = kapasitas
    def hitungharga(self):
        print('harga total : ',self.harga*self.jumlah)

class HardDiskSATA(ComputerPart):
    def __init__(self, nama,pabrikan, harga,kapasitas,rpm):
        super().__init__(nama, pabrikan,'SATA', harga)
        self.kapasitas = kapasitas
        self.rpm = rpm
    def hitungharga(self):
        print('harga total : ',self.harga*self.jumlah)

p = Prosesor('Intel','Core i 7 7740X',4350000,4,'4.3GHz')
m = RandomAccessMemory('V - Gen','DDR S0Dimm
PC19200/2400MHz',328000,'4GB')
hdd = HardDiskSATA('seagate','HDD 2.5 inc',295000,'500GB',7200)
parts = [p,m,hdd]
for i in parts:
    print('-'*25)
    print('{} {} produksi {} harga perbarang
{}'.format(i.jenis,i.nama,i.pabrikan,i.harga))
    i.hitungharga()
```

- Output

```
PS D:\KULIAH\Semester 2\Pemrograman Berorientasi Objek Praktik\Overloading&Overriding> & "C:/Users/ASUS Exp
indowsApps/python3.10.exe" "d:/KULIAH/Semester 2/Pemrograman Berorientasi Objek Praktik/Overloading&Overrid
masukan jumlah Intel: 7
masukan jumlah V - Gen: 4
masukan jumlah seagate: 5
------------------------
processor Intel produksi Core i 7 7740X harga perbarang 4350000
harga total :  30450000
------------------------
RAM V - Gen produksi DDR S0Dimm PC19200/2400MHz harga perbarang 328000
harga total :  1312000
------------------------
SATA seagate produksi HDD 2.5 inc harga perbarang 295000
harga total :  1475000
PS D:\KULIAH\Semester 2\Pemrograman Berorientasi Objek Praktik\Overloading&Overriding>
```

7. Polymorphism dengan class

- Source Code

```python
# Polymorphism dengan class

class Kucing:
    def __init__(self, nama, umur):
        self.nama = nama
        self.umur = umur

    def bersuara(self):
        print('Meow')

class Dog:
    def __init__(self, nama, umur):
        self.nama = nama
        self.umur = umur

    def bersuara(self):
        print('Guk...guk...')

kucing1 = Kucing("Tom", 3)
anjing1 = Dog("Spike", 4)

for hewan in (kucing1, anjing1):
    hewan.bersuara()
```

- Penjelasan
- Pada class kucing terdapat *def__init__*. Fungsi dari *__init__* yaitu melakukan inisialisasi pembuatan objek dari class. *Self* merupakan sebuah variable saja yang merujuk pada kelas itu sendiri. Dalam class kucing terdapat bersuara **Meow** dengan perintah print. Pada class Dog juga terdapat nama fungtion **bersuara** yang menampilkan output Guk… guk.

- Output

```
PS D:\KULIAH\Semester 2\Pemrograman Berorientasi Objek
indowsApps/python3.10.exe" "d:/KULIAH/Semester 2/Pemro
"
Meow
Guk...guk...
PS D:\KULIAH\Semester 2\Pemrograman Berorientasi Objek
```

8. Polymorphism dengan fungsi len

- Source Code

```
# Polymorphism
# Simple example using len function

print(len("polymorphism"))
print(len([0,1,2,3]))

# '''

# Menggunakan fungsi len
# Output:
# 12 (Tipe Data String)
# 4 (Tipe Data List)
# '''

# Using class
class jerman:
    def ibukota(self):
        print('Berlin adalah ibukota negara Jerman')

class jepang:
    def ibukota(self):
        print('Tokyo adalah ibukota jepang')

negara_1 = jerman()
negara_2 = jepang()

for country in (negara_1, negara_2):
    country.ibukota()
```

- Penjelasan
- Fungsi Print digunakan untuk menampilkan suatu program, *Len* digunakan untuk mengetahui panjang (jumlah item). Pada program diatas ada beberapa

class yang digunakan untuk mendefinisikan objek pada tiap class yang akan di tampilkan

- Output

```
PS D:\KULIAH\Semester 2\Pemrograman Berorientasi Objek Praktik\Overloading&Overriding> & "C:
indowsApps/python3.10.exe" "d:/KULIAH/Semester 2/Pemrograman Berorientasi Objek Praktik/Over
en.py"
12
4
Berlin adalah ibukota negara Jerman
Tokyo adalah ibukota jepang
PS D:\KULIAH\Semester 2\Pemrograman Berorientasi Objek Praktik\Overloading&Overriding>
```

9. Polymorphism dengan inheritance

- Source Code

```
# Polymorphism dengan inheritance

class burung:
    def intro(self):
        print("Di dunia ini ada beberapa type berbeda dari spesies
burung")

    def terbang(self):
        print("Hampir semua burung dapat terbang, namun ada beberapa
yang tidak dapat terbang")

class Elang(burung):
    def terbang(self):
        print("Elang dapat terbang")

class BurungUnta(burung):
    def terbang(self):
        print("Burung unta tidak dapat terbang")

obj_burung = burung()
obj_elang = Elang()
obj_burung_unta = BurungUnta()

obj_burung.intro()
obj_burung.terbang()

obj_elang.intro()
obj_elang.terbang()

obj_burung_unta.intro()
```

```
obj_burung_unta.terbang()
```

- Penjelasan

  Class burung digunakan untuk menampilkan output tentang burung. Selanjutnya class Elang memanggil data dari class burung yang digunakan untuk menyimpan output yang menjelaskan tentang elang. Kemuadian class burung unta digunakan untuk menampilkan output tentang buru unta. Selanjutnya pada setiap class akan di tampilkan atau di cetak.

- Output

```
PS D:\KULIAH\Semester 2\Pemrograman Berorientasi Objek Praktik\Overloading&Overriding> & "C:/Users/ASUS
indowsApps/python3.10.exe" "d:/KULIAH/Semester 2/Pemrograman Berorientasi Objek Praktik/Overloading&Over
nce.py"
Di dunia ini ada beberapa type berbeda dari spesies burung
Hampir semua burung dapat terbang, namun ada beberapa yang tidak dapat terbang
Di dunia ini ada beberapa type berbeda dari spesies burung
Elang dapat terbang
Di dunia ini ada beberapa type berbeda dari spesies burung
Burung unta tidak dapat terbang
PS D:\KULIAH\Semester 2\Pemrograman Berorientasi Objek Praktik\Overloading&Overriding>
```

10. Computerpart

- Source Code

```
class ComputerPart:
    def __init__(self, pabrikan, nama, jenis, harga):
        self.pabrikan = pabrikan
        self.nama = nama
        self.jenis = jenis
        self.harga = harga

class Processor(ComputerPart):
    def __init__(self, pabrikan, nama, harga, jumlah_core, speed):
        super().__init__(pabrikan, nama, 'processor', harga)
        self.jumlah_core = jumlah_core
        self.speed = speed

    # method overloading
    def kecepatanProcessor(self, speed):
        if(speed >= 3):
            print("Kecepatan Processor sangat cepat")
        else :
            print("Kecepatan processor normal")

class RandomAccessMemory(ComputerPart):
```

```
    def __init__(self, pabrikan, nama, harga, kapasitas):
        super().__init__(pabrikan, nama, 'RAM', harga)
        self.kapasitas = kapasitas

class HardDiskSATA(ComputerPart):
    def __init__(self, pabrikan, nama, harga, kapasitas, rpm):
        super().__init__(pabrikan, nama, 'SATA', harga)
        self.kapasitas = kapasitas
        self.rpm = rpm

p = Processor('Intel', 'Core i9', 4000000, 4,'4 Ghz')
m = RandomAccessMemory('Sandisk', 'DD4 4 SECEPAT KILAT', 800000,'32
GB')
hdd = HardDiskSATA('WD', 'WD Green', 1200000, '1000 GB',7200)

parts = [p,m,hdd]
for part in parts:
    print('{} {} pabrikan {}'. format(part.jenis,
part.nama,part.pabrikan))

p.kecepatanProcessor(3.5)
```

- Penjelasan

  Pada kelas Computerpart terdapat *def__init__*. Fungsi dari *__init__* yaitu melakukan inisialisasi pembuatan objek dari class. *Self* merupakan sebuah variable saja yang merujuk pada kelas itu sendiri. detComputer digunakan untuk mengetahui data pabrikan dan nama dari komputer.Hitungharga berfunsi untuk cari tau harga dari sebuah komputer.

- Output

```
PS D:\KULIAH\Semester 2\Pemrograman Berorientasi Objek Praktik\Overloading&Overriding> & "C:/
indowsApps/python3.10.exe" "d:/KULIAH/Semester 2/Pemrograman Berorientasi Objek Praktik/Overl
processor Core i9 pabrikan Intel
RAM DD4 4 SECEPAT KILAT pabrikan Sandisk
SATA WD Green pabrikan WD
Kecepatan Processor sangat cepat
PS D:\KULIAH\Semester 2\Pemrograman Berorientasi Objek Praktik\Overloading&Overriding>
```

4. Kesimpulan

   Pada setiap program diatas dapat disimpulkan bahwa polymorphism dapat mempersingkat penulisan suatu program. Overloading yaitu penggunaan nama

dalam satu method yang berbeda. Overriding terjadi ketika deklarasi method subclass dengan nama dan parameter yang sama dengan method superclass.