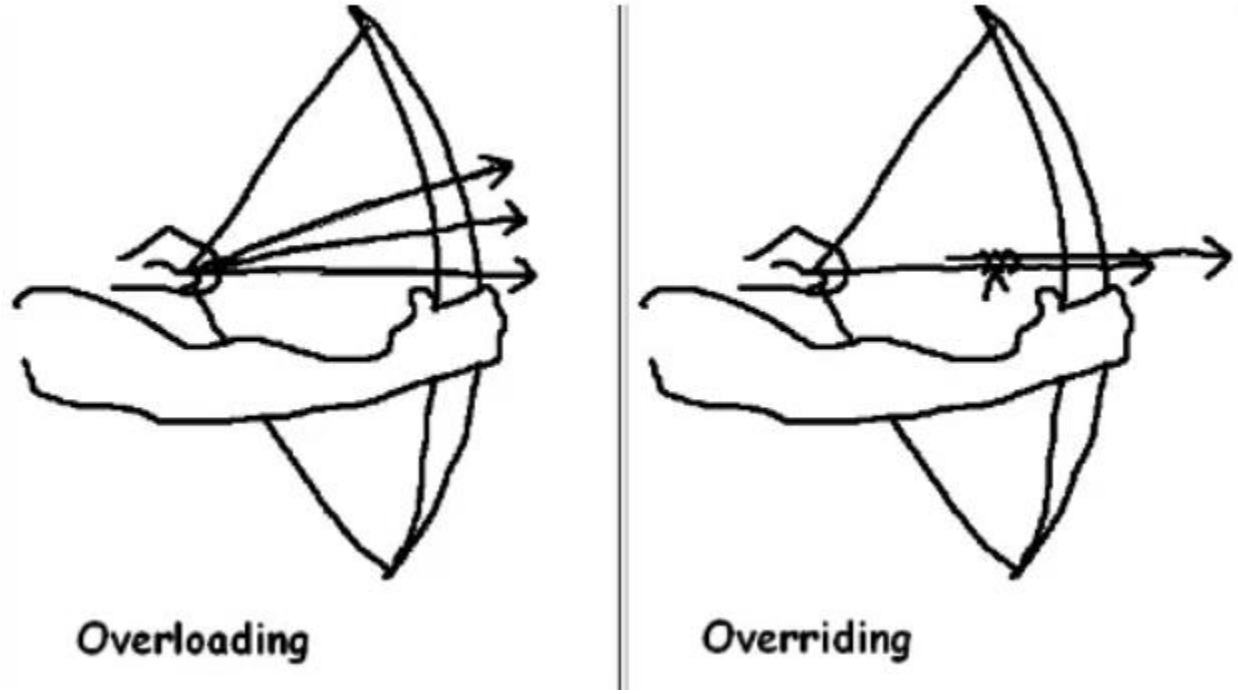
Pertemuan 6 (Praktikum)

POLYMORPHISM

KONSEP POLYMORPHISM

- Bentuk polymorphism ada beberapa diantaranya, yaitu polymorphism dengan fungsi, polymorphism dengan class, dan polymorphism dengan inheritance.
- Metode overloading merupakan metode yang berguna dalam situasi kompleks dengan penggunaan berbasis kondisi dan eksekusi parameter tertentu diperlukan. Metode ini harus digunakan secara tepat untuk menghindari penggunaan parameter yang salah.
- Metode overriding menunjukkan implementasi kelas yang sama dengan cara yang berbeda. Metode ini sangat berkaitan erat dengan *inheritance* (pewarisan), terutama kelas induk mewariskan metode dan atributnya ke beberapa kelas turunannya.
- Abstraction bertujuan untuk menyembunyikan detail yang tidakterlalu penting dari user. Python memiliki modul untuk menggunakan *Abstract Base Classes* (ABC).

Berikut ini adalah ilustrasi untuk memahami overloading dengan overriding:



Kode program untuk implementasi polymorphism dengan fungsi:

```python
# Polymorphism dengan Fungsi

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 negara Jepang")

negara1 = jerman()
negara2 = jepang()

for country in (negara1,negara2):
    country.ibukota()
```

Kode program untuk implementasi polymorphism dengan class:

```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()
```

Kode program untuk implementasi polymorphism dengan inheritance:

```
# 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()
```

Kode program untuk implementasi kelas abstrak:

```
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())
```

Kode program untuk implementasi overloading yaitu class Pegawai:

```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)

    # Method Overloading
    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)  # Overloading
peg2.tunjangan(2500)       # Overloading

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

Kode program untuk implementasi overriding yaitu class Segiempat:

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

    def hitungLuas(self):  # Method Overriding
        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):  # Method Overriding
        print("Luas Bujur sangkar =", self.side*self.side, "m^2")

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

Kode program untuk implementasi overloading yaitu:

```python
# Implementasi Overloading

class Mahasiswa:
    def __init__(self,nama,nim):
        self.nama = nama
        self.nim = nim
        #self.prodi = prodi
        #self.thn_masuk = thn_masuk
        #self.semester = semester

    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)

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

==========

TUGAS

Membuat laporan dengan format:

- Cover
- Isi (Teori+Tujuan+Diagram class+10 Kode program+Penjelasan+Output)
    1) Buatlah program untuk *Overloading* pada class ComputerPart
    2) Buatlah program untuk *Overriding* pada class ComputerPart
- Kesimpulan

Laporan dalam bentuk PDF (dikumpulkan di e-learning)

Program yang diupload ke github berjumlah 10