BAB 6
METHOD OVERRIDING

6.1 Maksud dan Tujuan

6.1.1 Maksud

Bab ini akan membahas tentang *method overriding* yang merupakan cara untuk kelas anak merubah *method* yang merupakan warisan dari kelas induk.

6.1.2 Tujuan

1. Mahasiswa memahami pengertian *method overriding*.

2. Mahasiswa memahami dan mampu membuat script *method overriding*.

6.2 Dasar Teori

Overriding adalah kemampuan sebuah kelas untuk merubah implementasi *method* yang diwariskan oleh kelas induk. *Overriding* merupakan hal yang penting dalam PBO karena kelas anak tidak kaku terhadap warisan yang diturunkan dari kelas induk. Dengan *method overriding*, kelas dapat melakukan banyak hal seperti menyalin sifat kelas lain, menghindari duplikasi kodingan, dan melakukan perubahan pada saat bersamaan. *Method overriding* merupakan hal penting dalam mekanisme pewarisan (*inheritance*).

Untuk melakukan *method overriding* pada Python sangat sederhana. Cukup membuat sebuah *method* pada keals anak dengan nama yang sama dengan *method* yang ada pada kelas induk. Ketika *method* sudah dibuat pada kelas anak maka *method* yang merupakan warisan kelas induk tidak akan dieksekusi ketika dipanggil.

```
class induk(object):
    def __init__ (self):
        self.nilai = 5

    def get_nilai(self):
        return self.nilai

class anak(induk):
    def get_nilai(self):
        return self.nilai + 1

anak1 = anak()
print(anak1.get_nilai())
```

Pada kodingan diatas terlihat bahwa *method* **get_nilai()** pada kelas anak telah dimodifikasi (tidak sama dengan *method* **get_nilai()** pada kelas induk. *Method* **get_nilai()** akan mengembalikan nilai **self.nilai = 5** jika tidak melakukan *overriding* terhadap *method* tersebut. Tetapi pada kodingan terlihat ada modifikasi yang dilakukan dan itu menyatakan *method overriding* sudah dilakukan sehingga *method* **get_nilai()** akan mengembalikan nilai **self.nilai = 5 + 1**.

6.3 Latihan

1. Salin kodingan kelas bangunRuang berikut dan buat sebuah obyek dari kelas bangunRuang dan tampilkan volume dari obyek tersebut!

```
import math


class bangunRuang(object):
    """docstring for bangunRuang"""
    def __init__(self):
        super(bangunRuang, self).__init__()
        self.nama = 'Bangun Ruang'
        self.luasa = None
        self.tinggi = None


    def volume(self, luasa, tinggi):
        self.volume = luasa*tinggi
        return self.volume
```

2. Buatlah kelas tabung, balok, dan bola sebagai kelas anak dari kelas bangunRuang! Gunakan perintah **math.pi** untuk menggunakan nilai **pi**.
3. Modifikasi program yang telah dibuat sehingga bisa melakukan inputan melalui keyboard!