

BAB VII. CLASS DAN OBJECT

7.1. Pengertian Class dan Object

7.1.1. Class

Class adalah struktur dasar dari OOP (Object Oriented Programming). Terdiri dari dua tipe yaitu : field (attribute/property) dan method (behavior). Class digunakan untuk mendeklarasikan sebuah variabel yang berupa objek atau dinamakan “referensi objek (*object reference*)”

1. Attribute

- Berlaku sebagai data, didefinisikan oleh class, individu, berbeda satu dengan lainnya.
- Menggambarkan tampilan, status, kualitas dari object.
- Contoh :
class motorcycle
 attribute-nya = color [red, green, silver]
 style [cruiser, sport bike, standart]
 make [Honda, BMW]
- Didefinisikan dalam class dengan menggunakan variabel.

2. Behavior

- Berlaku sebagai method (operasi).
- Menggambarkan bagaimana sebuah *instance class* beroperasi misal bagaimana reaksi dari *class* jika diminta untuk melakukan sesuatu hal.
- Contoh :
class motorcycle
 behavior-nya = start the engine
 stop the engine
 change gear
- Untuk menentukan *behavior* dari *object* harus membuat *Methods*.

Contoh Program :

```

class Motorcycle {
    String make;
    String color;
    boolean engineState;

void startEngine() {
    if (engineState == true)
        System.out.println("The engine is already on.");
    else {
        engineState = true;
        System.out.println("The engine is now on.");
    }
}

void showAtts() {
    System.out.println("This motorcycle is a "
        + color + " " + make);
    if (engineState == true)
        System.out.println("The engine is on.");
    else System.out.println("The engine is off.");
}

public static void main (String args[]) {
    Motorcycle m = new Motorcycle();
    m.make = "Yamaha RZ350";
    m.color = "yellow";
    System.out.println("Calling showAtts...");
    m.showAtts();
    System.out.println("-----");
    System.out.println("Starting engine...");
    m.startEngine();
    System.out.println("-----");
    System.out.println("Calling showAtts...");
    m.showAtts();
    System.out.println("-----");
    System.out.println("Starting engine...");
    m.startEngine();
}
}

```

7.1.2. Object

Setiap Object (obyek) dibangun dari sekumpulan data (atribut) yang disebut "variabel" (untuk menjabarkan karakteristik khusus dari obyek) dan juga terdiri dari sekumpulan *method* (menjabarkan tingkah laku dari obyek) atau Obyek adalah = sebuah perangkat lunak yg berisi sekumpulan variabel dan method yg berhubungan. Obyek mrpk.sebuah instance (keturunan) dari class. Variabel dan method diketahui sbg. variabel instance dan method instance.

7.2. Instansiasi Class dan Object

1. Mendefinisikan kelas baru

```
class Kotak{
    double panjang;
    double lebar;
    double tinggi; }
```



Pendefinisian class hanya akan membuat pola (template) dan tipe data baru bukan objek baru.

2. Mendeklarasikan Objek

```
Kotak k ;
k= new Kotak( )
```



Atau

```
Kotak k= new Kotak( )
```

- Deklarasikan variabel yang digunakan sebagai referensi ke objek yang bersangkutan.
- Menginstansiasi class dengan menggunakan operator "new"
- Melalui objek k dapat mengakses dan manipulasi data.

3. Mengakses Data

```
k.panjang = 4;
k.lebar = 3;
k.tinggi = 2;
```



Mengisikan data dari class kotak yaitu panjang, lebar dan tinggi

Program lengkapnya

```

1 //PROGRAM 7-1
2
3 class Kotak {
4     double panjang;
5     double lebar;
6     double tinggi;
7 }
8
9 class DemoKotak1 {
10     public static void main(String[] args) {
11
12         double volume;
13         Kotak k = new Kotak();
14
15         // Mengisikan nilai ke dalam data-data kelas Kotak
16         k.panjang = 4;
17         k.lebar = 3;
18         k.tinggi = 2;
19
20         // Menghitung isi/volume kotak
21         volume = k.panjang * k.tinggi * k.lebar;
22
23         // Menampilkan nilai volume ke layar monitor
24         System.out.println("Volume kotak = " + volume);
25     }
26 }
27

```

Program di atas harus disimpan dengan nama **DemoKotak1.java** bukan Kotak.java karena *method main* () terletak di class DemoKotak1. Pada saat kompilasi program akan membentuk 2 buah file class, yaitu **Kotak.class** dan **DemoKotak1.class**.

Setiap objek dari class akan memiliki salinan data sendiri-sendiri, artinya antara objek satu dengan lainnya dapat mempunyai nilai data yang berbeda.

Bandingkan contoh program di bawah:

Program memiliki 2 objek yang masing-masing bernama k1 dan k2.

```
1 //PROGRAM 7-2
2
3 class Kotak {
4     double panjang;
5     double lebar;
6     double tinggi;
7 }
8
9 class DemoKotak2 {
10     public static void main(String[] args) {
11
12         double volume1, volume2;
13
14         Kotak k1 = new Kotak(); // mendeklarasikan objek k1
15         Kotak k2 = new Kotak(); // mendeklarasikan objek k2
16
17         // Mengisikan nilai ke dalam objek k1
18         k1.panjang = 4;
19         k1.lebar = 3;
20         k1.tinggi = 2;
21
22         // Mengisikan nilai ke dalam objek k2
23         k2.panjang = 6;
24         k2.lebar = 5;
25         k2.tinggi = 4;
26
27
28         // Menghitung isi/volume dari objek k1
29         volume1 = k1.panjang * k1.tinggi * k1.lebar;
30
31         // Menghitung isi/volume dari objek k2
32         volume2 = k2.panjang * k2.tinggi * k2.lebar;
33
34         // Menampilkan nilai volume k1 dan k2 ke layar monitor
35         System.out.println("Volume k1 = " + volume1);
36         System.out.println("Volume k2 = " + volume2);
37     }
38 }
39
```

Contoh berikut merupakan contoh deklarasi dari class Orang dengan objek adalah O.

```
//PROGRAM 7-3
```

```
class Orang{
    String nama;
    String alamat;
    String no_telp;
};

class CobaOrang {
    public static void main(String[] args) {
        Orang O = new Orang();

        // Mengisikan nilai ke dalam data-data kelas Orang
        O.nama = "Putri Mahadewi";
        O.alamat = "Jl. Pemuda 113 Magelang" ;
        O.no_telp = "0293-360345";

        //Menampilkan data Orang
        System.out.println("DATA ORANG");
        System.out.println("=====");
        System.out.println("NAMA = "+O.nama);
        System.out.println("ALAMAT = "+O.alamat);
        System.out.println("NO.TELP = "+O.no_telp);
    }
}
```

7.3. Method

7.3.1. Mendefinisikan Method


Berdasarkan contoh program di atas, berikut method yang ditambahkan, yaitu method untuk mencetak Volume.

```
void cetakVolume() {  
    System.out.println("Volume kotak = " +  
        (panjang * lebar * tinggi));}
```

Semua *method* dalam suatu *class* dapat mengakses data-datanya secara langsung tanpa melalui referensi. Pemanggilan method dilakukan dengan menuliskan objek pemiliknya dan diikuti oleh operator titik (.) beserta nama method yang akan dieksekusi.

Untuk memanggil method adalah:

```
Objek ke-1.nama_method;  
Objek ke-2.nama_method;  
Objek ke-3.nama_method;  
Objek ke-n.nama_method;
```



```
k1.cetakVolume;  
k2.cetakVolume;  
k3.cetakVolume;
```

Program lengkapnya:

```

1 //PROGRAM 7-4
2
3 class Kotak {
4     double panjang;
5     double lebar;
6     double tinggi;
7
8     // Mendefinisikan method void (tidak mengembalikan nilai)
9     void cetakVolume() {
10        System.out.println("Volume kotak = " + (panjang * lebar * tinggi));
11    }
12 }
13 class DemoMethod1 {
14     public static void main(String[] args) {
15         Kotak k1, k2, k3;
16
17         // instansiasi objek
18         k1 = new Kotak();
19         k2 = new Kotak();
20         k3 = new Kotak();
21
22         // mengisi data untuk objek k1
23         k1.panjang = 4;
24         k1.lebar = 3;
25         k1.tinggi = 2;
26
27         // mengisi data untuk objek k2
28         k2.panjang = 6;
29         k2.lebar = 5;
30         k2.tinggi = 4;
31
32         // mengisi data untuk objek k3
33         k3.panjang = 8;
34         k3.lebar = 7;
35         k3.tinggi = 6;
36
37         // memanggil method cetakVolume() untuk masing-masing objek
38         k1.cetakVolume();
39         k2.cetakVolume();
40         k3.cetakVolume();
41     }
42 }
43

```

Untuk method dalam proses perhitungan (mengembalikan nilai), contoh akan menghitung volume. Tambahkan statemen di bawah ini ke dalam listing program 7-4.

```

double hitungVolume() {
    double vol = panjang * lebar * tinggi;
    return vol;
}

```

Latihan

- Modifikasilah program 7-4 dengan melengkapi method hitungVolume dan input data.
- Lengkapi program di bawah.


```
class {
    nama;
    int umur;
    Person(String n, int a) {
        nama = n;
        umur = a;
    }
    void printPerson() {
        System.out.print("Hallo...Aku " + name);
        System.out.println("Aku " + umur + " tahun");
    }
    public static void main (String args[]) {
        p.printPerson();
        System.out.println("-----");
        p.printPerson();
        System.out.println("-----");
    }
}
```

7.3.2. Overload Terhadap Method

Dalam suatu kelas, dapat mendefinisikan banyak method dengan nama yang sama, selama parameter yang terdapat pada method-method tersebut berbeda. Parameter dalam method dikatakan berbeda dari method yang lain apabila:

- a) Jumlahnya berbeda, walaupun tipe datanya sama.
- b) Tipe datanya berbeda, walaupun jumlahnya sama.
- c) Jumlah dan tipe datanya berbeda.
- d) Urutan parameter berbeda, walaupun jumlah dan tipe datanya sama.

Proses pendefinisian method dengan nama sama disebut dengan “*overload*”.

```
int bagi (int a, int b)
{
return a/b;
}
```

```
double bagi (double a, double b)
{
return a/b;
}
```

→ Nama sama tipe data
berbeda → benar

```
int bagi (int a, int b)
{
return a/b;
}
```

```
double bagi (int a, int b)
{
return a/b;
}
```

→ Duplikasi nama
method dan tipe
data → salah

```
void ketik(int a, String b)
{
System.out.print("int : " +a+",String:\\" +b+"\");
}
```

```
void ketik (String b, int a)
{
System.out.print("String:\\" +b+"\", int:" +a);
}
```

→ Urutan
parameter
berbeda → benar

Cobalah listing program berikut.

```
class Pembagian {
    int bagi(int a, int b) {
        return a/b;
    }

    double bagi(double a, double b) {
        return a/b;
    }
}

class DemoOverload1 {
    public static void main(String[] args) {

        Pembagian b = new Pembagian();

        int x = b.bagi(10, 4);
        double y = b.bagi(10.0, 4.0);

        System.out.println("Hasil bagi tipe int    = " + x);
        System.out.println("Hasil bagi tipe double = " + y);
    }
}
```

7.4. Constructor

7.4.1. Mendefinisikan Constructor

Constructor adalah *method* khusus yang didefinisikan di dalam kelas dan akan dipanggil secara otomatis setiap kali terjadi instansiasi objek, dan merupakan *method* yang mengembalikan tipe kelas (dirinya sendiri). Fungsi dari *constructor* adalah untuk melakukan instansiasi nilai terhadap data-data yang terdapat pada kelas bersangkutan. Apabila tidak mendefinisikan *constructor* maka secara otomatis Java akan membuatnya untuk kita. *Constructor* semacam ini disebut dengan “*default constructor*”, yang akan menginisialisasikan semua data yang ada dengan nilai nol, string dengan nilai null, variabel boolean diset ke *false*.

Beberapa hal yang perlu diperhatikan pada saat mendefinisikan *constructor* kelas adalah *constructor* tidak mempunyai tipe kembalian, nama *constructor* harus sama persis dengan nama kelas yang didefinisikan.

Program 7-1. DemoConstructor1.java

```
class Kotak {
    double panjang;
    double lebar;
    double tinggi;

    // Mendefinisikan constructor untuk kelas Kotak
    Kotak() {
        panjang = 4;
        lebar = 3;
        tinggi = 2;
    }

    double hitungVolume() {
        return (panjang * lebar * tinggi);
    }
}
```

```
class DemoConstructor1 {
    public static void main(String[] args) {

        Kotak k1, k2;

        k1 = new Kotak();
        k2 = new Kotak();

        System.out.println("Volume k1 = " + k1.hitungVolume());
        System.out.println("Volume k2 = " + k2.hitungVolume());
    }
    double hitungVolume() {
        return (panjang * lebar * tinggi);
    }
}

class DemoConstructor1 {
    public static void main(String[] args) {

        Kotak k1, k2;

        k1 = new Kotak();
        k2 = new Kotak();

        System.out.println("Volume k1 = " + k1.hitungVolume());
        System.out.println("Volume k2 = " + k2.hitungVolume());
    }
}
```

Pada program 7-1 di atas, ada proses inialisasi nilai panjang, lebar dan tinggi masing-masing adalah 4, 3, 2. yang akan berlaku pada semua objek kotak.

Untuk membuat Constructor berlaku dinamis, maka harus diberi parameter, seperti berikut.

Program 7-2. DemoConstructor2.java

```

1
2 class Kotak {
3     double panjang;
4     double lebar;
5     double tinggi;
6
7     // Mendefinisikan constructor dengan parameter
8     Kotak(double p, double l, double t) {
9         panjang = p;
10        lebar = l;
11        tinggi = t;
12    }
13
14    double hitungVolume() {
15        return (panjang * lebar * tinggi);
16    }
17 }
18
19 class DemoConstructor2 {
20     public static void main(String[] args) {
21
22         Kotak k1, k2;
23
24         k1 = new Kotak(4, 3, 2);
25         k2 = new Kotak(6, 5, 4);
26
27         System.out.println("Volume k1 = " + k1.hitungVolume());
28         System.out.println("Volume k2 = " + k2.hitungVolume());
29     }
30 }
31

```

7.4.2. Overload pada Constructor

Berdasarkan contoh class Kotak di atas, maka dapatlah dibuat overload pada constructor. Kasusnya adalah membuat tiga buah constructor yaitu constructor tanpa parameter, constructor yang memiliki satu buah parameter, dan constructor yang memiliki tiga buah parameter.

```

Kotak ( ){
    panjang = 0;
    lebar = 0;
    tinggi = 0;
}

```

→ Definisi constructor
tanpa parameter

```
Kotak (double sisi){
    panjang = lebar = tinggi = sisi;
}
```

Definisi constructor
dengan satu parameter

```
Kotak (double p,double l, double t){
    panjang = p;
    lebar = l;
    tinggi = t;
}
```

Definisi constructor
dengan tiga parameter

Lengkapilah program di bawah dengan menambahkan overload constructor di atas.

Program 7-3. DemoOverloadConstructor.java

```
// Deklarasi Class Kotak
// Mendefinisikan constructor tanpa parameter
// Mendefinisikan constructor dengan satu parameter
// Mendefinisikan constructor dengan tiga parameter
double hitungVolume() {
    return (panjang * lebar * tinggi); }
}
class DemoOverloadConstructor {
    public static void main(String[] args) {
        Kotak k1, k2, k3;
        k1 = new Kotak();
        k2 = new Kotak(10);
        k3 = new Kotak(4, 3, 2);
        //Menampilkan volume dari masing-masing objek Kotak
        System.out.println("Volume k1 = " + k1.hitungVolume());
        System.out.println("Volume k2 = " + k2.hitungVolume());
        System.out.println("Volume k3 = " + k3.hitungVolume());
    }
}
```