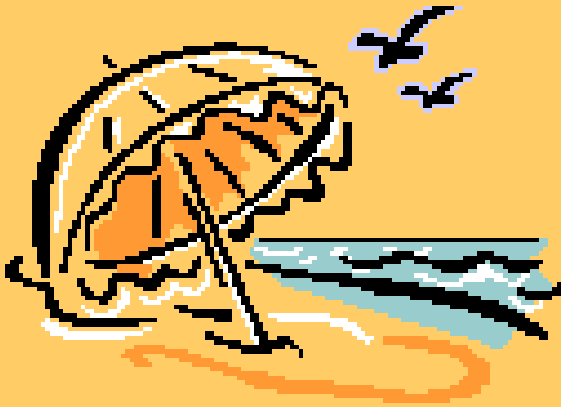
Pemrograman Berorientasi Objek / *Object Oriented Programming* / (OOP)



Nur Hasanah, M.Cs

- Object Oriented Programming (OOP) adalah inti dari pemrograman Java.
- Dalam OOP, setiap **objek** didefinisikan sebagai suatu entitas yang memiliki **data** dan **method**.
- Data disebut juga sifat / variabel / konstanta sedangkan method adalah perilaku / kemampuan melakukan sesuatu / fungsi.
- Contoh: manusia adalah suatu objek yang memiliki data berupa nama, jenis kelamin, tinggi badan, berat badan, dsb), dan juga method berupa cara bicara, cara berjalan, cara marah, dsb.

- **Kelas** adalah bentuk **abstrak** dari suatu objek.
- Wujud nyata dari suatu kelas adalah disebut ***instance***.
- Contoh: apabila terdapat kelas Manusia, maka contoh instancenya (objek) adalah : Udin, Kabayan, dll.
- Contoh lain: apabila terdapat kelas Kucing, maka contoh instancenya (objek) adalah : Si Meong, Si Manis, Si PusPus, dsb.

Ciri-ciri OOP

- **Pembungkusan (*Encapsulation*)**

  Membungkus semua kode dan data yang berkaitan ke dalam satu entitas tunggal (objek). Pembungkusan menggunakan acces modifier seperti ***private***, ***protected***, ***public***.
- **Pewarisan (*Inheritance*)**

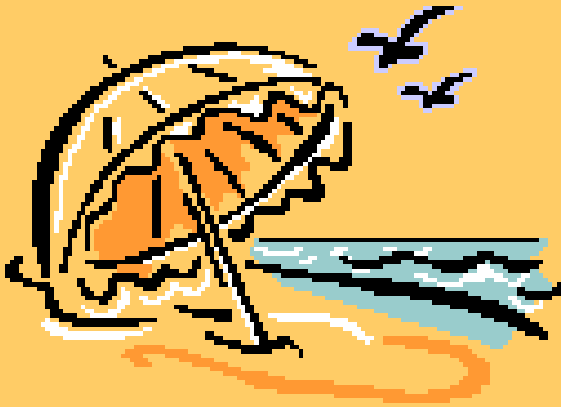
  Suatu kelas dapat diturunkan menjadi kelas-kelas baru lainnya (*subclass*) yang mewarisi beberapa sifat atau perilaku kelas induknya (*superclass)*.

Ciri-ciri OOP

- **Polimorfisme (*Polymorfism*)**

  Kemampuan suatu objek untuk mengungkap banyak hal melalui cara yang sama.

Kelas dan Objek



- Kelas dapat didefinisikan sebagai cetak biru (*blueprint*) / prototipe / kerangka yang mendefinisikan variabel-variabel (data) dan method-method (perilaku) dari objek tertentu.
- Kelas adalah pola (*template*) untuk pembuatan objek, dan objek adalah wujud nyata (instance) dari sebuah kelas.

Mendeklarasikan Objek

1. Medeklarasikan variabel yang digunakan sebagai referensi ke objek dari kelas yang bersangkutan.

```
Kotak k;
```

2. Menginstantiasi kelas dengan menggunakan operator new dan memasukkan instancenya ke dalam variabel referensi yang baru saja dideklarasikan.

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

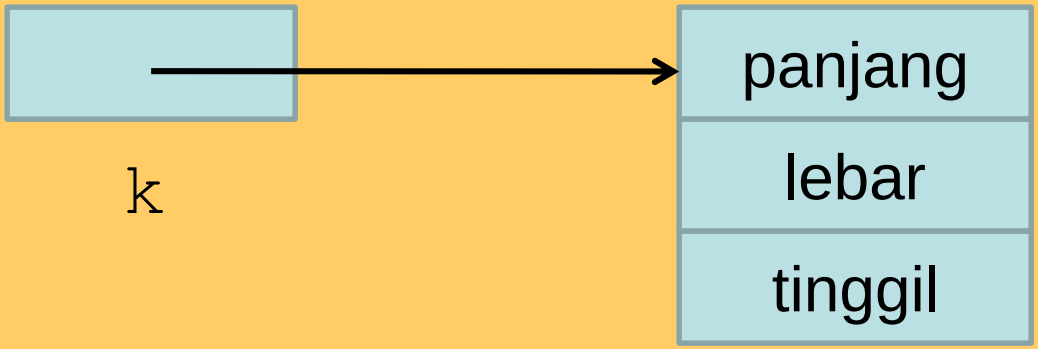
Atau biasanya ditulis satu baris:

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

Mendeklarasikan Objek

```
Kotak k;
```


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



Objek `Kotak`

Contoh Program : DemoKotak1.java

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

class DemoKotak1 {
    public static void main(String[] args) {

    double volume;
    Kotak k = new Kotak();
    k.panjang = 4;
    k.lebar = 3;
    k.tinggi = 2;
    volume = k.panjang * k.tinggi * k.lebar;
    System.out.println("Volume kotak = " +
       volume);
  }
}
```

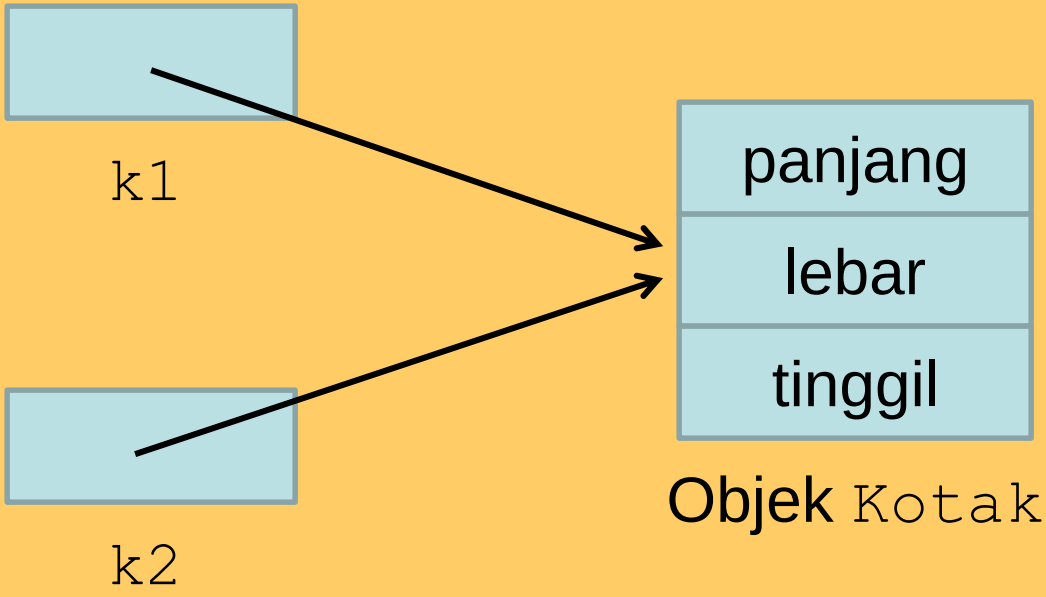
```
class Kotak {
  double panjang;
  double lebar;
  double tinggi;
}

class DemoKotak2 {
public static void main(String[] args)
{

    double volume1, volume2;

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

    k1.panjang = 4;
    k1.lebar = 3;
    k1.tinggi = 2;

    k2.panjang = 6;
    k2.lebar = 5;
    k2.tinggi = 4;

    volume1 = k1.panjang * k1.tinggi * k1.lebar;
    volume2 = k2.panjang * k2.tinggi * k2.lebar;

System.out.println("Volume k1 = " + volume1);
System.out.println("Volume k2 = " + volume2);
  }
}
```

Contoh Program : DemoKotak2.java

Mengisi Nilai pada Referensi Objek : DemoReferensi1.java

```
class Kotak {
  double panjang;
  double lebar;
  double tinggi;
}
class DemoReferensi1 {
  public static void main(String[] args) {
    double volume1, volume2;
    Kotak k1, k2;
    k1 = new Kotak();
    k2 = k1;
    k1.panjang = 4;
    k1.lebar = 3;
    k1.tinggi = 2;
    volume1 = k1.panjang * k1.tinggi * k1.lebar;
    volume2 = k2.panjang * k2.tinggi * k2.lebar;
    System.out.println("Volume k1 = " + volume1);
    System.out.println("Volume k2 = " + volume2);
  }
}
```

Mengisi Nilai pada Referensi Objek : DemoReferensi1.java



Mengisi Nilai pada Referensi Objek : DemoReferensi2.java

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

class DemoReferensi2 {
  public static void main(String[] args) {
    double volume1, volume2;
    Kotak k1, k2;
    k1 = new Kotak();
    k2 = k1;
    k1.panjang = 4;
    k1.lebar = 3;
    k1.tinggi = 2;
    volume1 = k1.panjang * k1.tinggi * k1.lebar;
    volume2 = k2.panjang * k2.tinggi * k2.lebar;
    System.out.println("Sebelum k1 diubah:");
    System.out.println("Volume k1 = " + volume1);
    System.out.println("Volume k2 = " + volume2);
```

```
    k1 = new Kotak();
    k1.panjang = 6;
    k1.lebar = 5;
    k1.tinggi = 4;

    volume1 = k1.panjang * k1.tinggi * k1.lebar;
    volume2 = k2.panjang * k2.tinggi * k2.lebar;
    System.out.println("\nSetelah k1 diubah:");
    System.out.println("Volume k1 = " + volume1);
    System.out.println("Volume k2 = " + volume2);
  }
}
```

Constructor

- Constructor adalah method khusus yang didefinisikan di dalam kelas dan akan dipanggil secara otomatis setiap kali terjadi instantiasi objek.
- Constructor berfungsi melakukan inisialisasi nilai terhadap data-data pada kelas yang bersangkutan.
- Apabila kita tidak mendefiniskannya, Java akan membuatkkannya secara otomatis.
- Default constructor menginisialisasi semua data dengan nilai nol.

Constructor

- Namun bila kita mendefinisikan constructor baru, maka default constructor sudah tidak berfungsi lagi.
- Sama dengan method, constructor dapat memiliki parameter dan dapat di-**overload**.
- Perlu diingat, constructor tidak memiliki nilai kembalian, tidak juga **`void`**.
- Nama constructor harus sama persis dengan nama kelas yang didefinisikan.

Contoh Program :
DemoConstructor1.java

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

  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());
  }
}
```

Contoh Program :
DemoConstructor2.java

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

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

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

```
class DemoConstructor2 {
  public static void main(String[] args) {

    Kotak k1, k2;

    k1 = new Kotak(4, 3, 2);
    k2 = new Kotak(6, 5, 4);

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

Kata Kunci ***This***

- **`This`** merupakan referensi ke objek yang sedang aktif.
- **`This`** digunakan di dalam method untuk mewakili nama kelas yang bersangkutan.

Contoh penggunaan ***This***

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

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

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

  Kotak(double panjang, double lebar, double tinggi)
    {
    this.panjang = panjang;
    this.lebar = lebar;
    this.tinggi = tinggi;
  }
```

Overload pada *Constructor :*
DemoOverloadConstructor.java

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

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

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

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

```
    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);

    System.out.println("Volume k1 = " + k1.hitungVolume());
    System.out.println("Volume k2 = " + k2.hitungVolume());
    System.out.println("Volume k3 = " + k3.hitungVolume());
  }
}
```

Objek sebagai Parameter

- Objek dapat digunakan sebagai parameter pada method. Contoh : **DemoParamObjek1.java**
- Objek dapat juga digunakan sebagai parameter pada constructor. Contoh : **DemoParamObjek2.java**

Meningkatkan tingkat Akses Data dan Method

- Dalam Pembungkusan (*Encapsulation*), kita menggabungkan data dan kode menjadi satu.
- Pada situasi seperti ini, kita dapat menetukan tingkat akses data dan method.
  - **Private** : data dan method hanya dapat diakses oleh kelas yang memilikinya.
  - **Protected** : data dan method dapat diakses oleh kelas yang memilikinya dan kelas-kelas turunannya.
  - **Public** : data dan method dapat diakses oleh kelas yang memilikinya, kelas-kelas turunannya dan semua kelas dari lingkungan luar.
  - **Default** : data dan method dapat diakses oleh kelas yang berada dalam satu paket.

Contoh : DemoPublicdanPrivate.java

```
class TingkatAkses {
  int a;
  public int b;
  private int c;

  public void setC(int nilai) {
    c = nilai;
  }

  public int getC() {
    return c;
  }
}
```

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

    TingkatAkses obj = new TingkatAkses();

    obj.a = 10; // BENAR, karena a secara default bersifat public
    obj.b = 20; // BENAR, karena b bersifat public

    //obj.c = 30; // SALAH, karena c bersifat private

    obj.setC(30); // BENAR, karena method setC() bersifat public

    System.out.println("Nilai obj.a : " + obj.a);
    System.out.println("Nilai obj.b : " + obj.b);
    System.out.println("Nilai obj.c : " + obj.getC());
  }
}
```

Kata Kunci ***`static`***

- Java mengizinkan kita untuk mengakses suatu anggota kelas (data atau method) tanpa harus membuat objeknya terlebih dahulu.
- Caranya, kita harus menjadikan data atau method tersebut bersifat statis, dengan kata kunci **`static`** pada awal deklarasi.

Kata Kunci ***static***

Terdapat batasan-batasan untuk method statis:

- Method statis hanya dapat memanggil method yang bersifat statis.
- Method statis hanya dapat mengakses data-data yang bersifat statis.
- Method statis tidak dapat diacu melalui referensi **`this`** maupun **`super`** (**`super`** dibahas di pertemuan berikutnya)

Contoh penggunaan ***Static*** :
DemoStatic3.java

```
class DeklarasiStatik {
  static int a;
  static int b;

  static void test() {
    int c = a + b;
    System.out.println("a + b = " + c);
  }
}

class DemoStatik3 {
  public static void main(String[] args) {
    DeklarasiStatik.a = 10;
    DeklarasiStatik.b = 20;
    DeklarasiStatik.test();
  }
}
```

Kata Kunci ***final***

3 fungsi kata kunci final :

- Apabila digunakan untuk mendeklarasikan variabel, maka nilai dari variabel tersebut tidak dapat diubah (diperankan sebagai konstanta).
- Apabila digunakan untuk mendeklarasikan method, maka method tersebut tidak dapat di-*override* oleh kelas-kelas turunannya.
- Apabila digunakan untuk mendefinisikan kelas, maka kelas tersebut sudah tidak dapat diturunkan lagi.

* *Override* method dan turunan kelas dibahas pertemuan berikutnya.

Reference

- Budi Rahardjo dkk. (2012). "*Mudah Belajar Java*". Penerbit Informatika Bandung.