PRAKTEK PEMROGRAMAN 2

PEMROGRAMAN BERORIENTASI OBJEK *OBJECT ORIENTED PROGRAMMING* (OOP)

MUH. IZZUDDIN MAHALI, M.CS.

PT. Elektronika FT UNY

- 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

PT. Elektronika FT UNY
Muh. Izzuddin Mahali, M.Cs.

- 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();
```

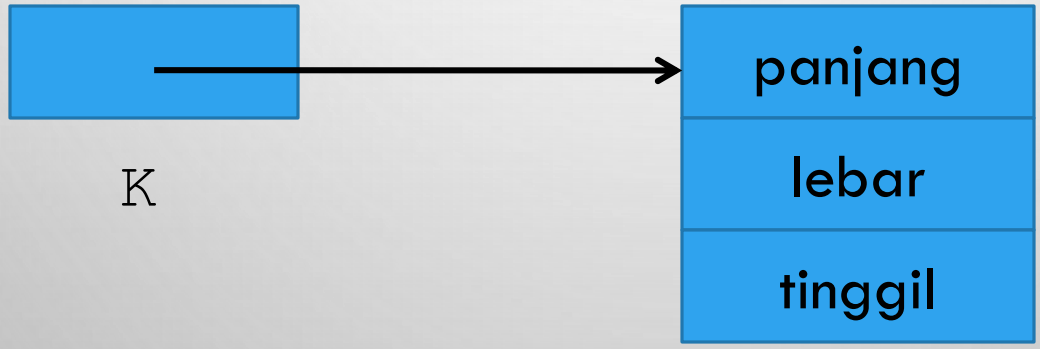
ATAU BIASANYA DITULIS SATU BARIS:

```
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;
```

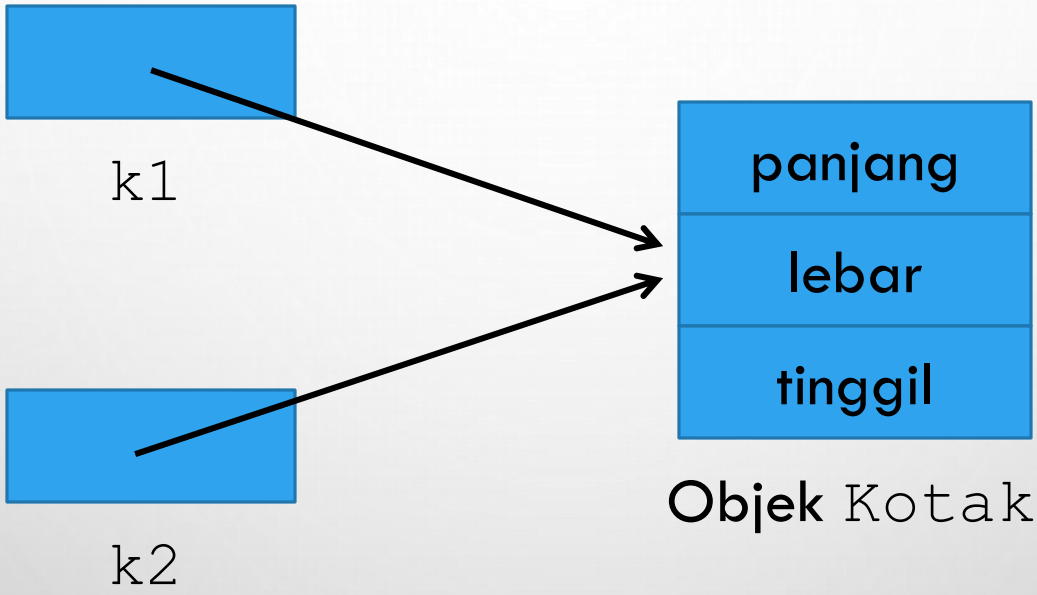
```
    k2.panjang = 6;
    k2.lebar = 5;
    k2.tinggi = 4;

    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 KHUSUSYANG 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 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.

SELESAI