MODUL 6
CONSTRUCTOR

6.1 Constructor

Constructor adalah method yang secara otomatis dipanggil/dijalankan pada saat sebuah class diinstansiasi. Jika dalam sebuah class tidak terdapat constructor maka secara otomatis Java akan membuatkan sebuah default constructor. Nama constructor harus sama dengan nama class dan tidak boleh memiliki tipe return value.

Sama halnya dengan method, constructor dapat memiliki satu atau banyak parameter maupun tanpa parameter. Constructor biasanya digunakan untuk memberi nilai awal dari atribut- atribut class tersebut.

Keyword "***super***" digunakan oleh subclass untuk memanggil constructor, atribut dan method yang ada pada superclass-nya.

Contoh untuk memanggil constructor milik superclass-nya :

```
super()
super(parameter)
```

Contoh untuk memanggil atribut dan method milik superclass-nya :

```
super.namaAtribut
super.namaMethod(parameter)
```

6.2 Static Method dan Static Property vs Variable Instant

Static property dan static method adalah property (variabel) dan method (function) yang melekat kepada class, bukan kepada objek. Konsep static property memang 'agak keluar' dari konsep objek sebagai tempat melakukan proses, karena sebenarnya class hanya merupakan '*blueprint*' saja.

Untuk membuat static property dan static method, kita menambahkan keyword ***'static'*** setelah penulisan akses level property atau method, seperti contoh berikut:

```
public static String pemilik;

public static void keterangan  pemilik(String pemilik) {..... }
```

Dalam contoh diatas menggunakan hak akses public, tetapi kita juga bisa menggunakan hak akses lain seperti private dan protected untuk static property dan static method.

Karena static property dan static method adalah milik class, maka kita tidak perlu membuat objek untuk mengaksesnya, tapi langsung menyebutkan nama class, berikut adalah contoh pengaksesan static property dan static method dari sebuah class.

```
ConstructorMotor.pemilik = "Ahmad Afif";
ConstructorMotor.keterangan_pemilik(ConstructorMotor.pemilik);
```

Variabel-variabel yang dideklarasikan dengan tidak menggunakan keyword "*static*", maka variabel tersebut disebut dengan instant variabel (atau variabel instant).

a) Jika sebuah variable merupakan variable instant, maka masing-masing objek dari class tersebut akan memiliki variable yang sama dengan variable instant tersebut, perubahan nilai yang terjadi pada variable instant di satu objek tidak akan berpengaruh pada variable instant di objek yang berbeda.
b) Jika sebuah variable merupakan variable static (pada suatu class), maka variabel static tersebut adalah variabel yang sama di semua objek dari class tersebut. Sehingga perubahan nilai pada variabel static tersebut di suatu objek akan berpengaruh juga terhadap objek yang lainnya.
c) Nilai suatu variabel static akan selalu sama untuk semua *instant of class* (atau objek) dari sebuah class.

Sebuah variable dideklarasikan static apabila variable tersebut bersifat global bagi semua objek dari suatu class. Contohnya adalah variable yang menyimpan nilai jumlah objek yang telah dibuat.

Silahkan coba source code dibawah ini :

ConstructorKendaraan.java

```java
public class ConstructorKendaraan {
    private String merk;  // ini adalah instant variable
    private static String pemilik; // ini adalah static variable

    // ini adalah constructor tanpa parameter
    protected ConstructorKendaraan() {
        merk = null;
    }

    // overloading constructor
    // ini adalah constructor dengan parameter merk
    protected ConstructorKendaraan(String merk) {
        this.merk = merk;
        merk = null;
    }

    protected void setMerk(String merk) {
        this.merk = merk;
    }

    protected String getMerk() {
        return merk;
    }

    // ini adalah static method
    protected static void setPemilik(String pemilik) {
        ConstructorKendaraan.pemilik = pemilik;
    }

    // ini adalah static method
    protected static String getPemilik() {
        return ConstructorKendaraan.pemilik;
    }

    protected void tampil(String a)
    {   System.out.println(a);
        a = null;
    }

    protected void hapus()
    {   // menghapus variable private dari memory
        merk = null;
        pemilik = null;
    }
}
```

ConstructorMotor.java

```java
// ConstructorMotor turunan dari class ConstructorKendaraan
public class ConstructorMotor extends ConstructorKendaraan {
    private String warna; // ini adalah instant variable

    // ini adalah constructor dengan parameter merk & warna
    protected ConstructorMotor(String merk,String warna){
        // memanggil constructor parent (class ConstructorKendaraan)
        // dengan keyword super dan parameter merk
        super(merk);

        this.warna = warna;

        // menghapus variable parameter dari memory
        merk = null;
        warna = null;
    }

    protected String getWarna() {
        return warna;
    }

    protected void hapus()
    {   // menghapus variable private dari memory
        warna = null;
        // memanggil method hapus class parent (class ConstructorKendaraan)
        // dengan keyword super
        super.hapus();
    }
}
```

MainConstructorMotor.java

```java
public class MainConstructorMotor {
    public static void main(String args[])
    {   String pemilik = "Ahmad Afif";
        String merk = "Honda";
        String warna = "Merah";

        // cara akses static variable dan static method dapat dengan
        // memanggil class (ConstructorKendaraan) secara langsung
        ConstructorKendaraan.setPemilik(pemilik);
        System.out.println("Pemilik Kendaraan = "+ConstructorKendaraan.getPemilik());
        System.out.println("====================================");

        // variable merk menjadi parameter Constructor pada saat
        // instansiasi/membuat objek baru (ob)
        ConstructorKendaraan ob = new ConstructorKendaraan(merk);
        ob.tampil("Merk Kendaraan = "+ob.getMerk());
        // cara akses static variable dan static method dapat juga dengan
        // objek (ob) pada method getPemilik()
        ob.tampil("Pemilik Kendaraan = "+ob.getPemilik());
        System.out.println("====================================");

        // instansiasi/membuat objek baru (ob2) tanpa parameter
        ConstructorKendaraan ob2 = new ConstructorKendaraan();
        // bandingkan akses untuk menampilkan nilai pada instant variable (merk)
        // dan static variable (pemilik) melalui method getter setelah membuat
        // objek baru "ob2", dimana sebelumnya juga sudah membuat objek "ob".
        // instant variable (merk) nilainya akan hilang,
        // sedangkan static variable (pemilik) nilainya tidak hilang/berubah
        ob2.tampil("Merk Kendaraan (instant variable) = "+ob2.getMerk());
        ob2.tampil("Pemilik Kendaraan (static variable) = "+ob2.getPemilik());
        System.out.println("====================================");

        // variable merk dan warna menjadi parameter Constructor pada saat
        // instansiasi/membuat objek baru (ob3)
        ConstructorMotor ob3 = new ConstructorMotor(merk,warna);
        ob3.tampil("Merk Motor = "+ob3.getMerk());
        ob3.tampil("Warna Motor = "+ob3.getWarna());
        ob3.tampil("Pemilik Motor = "+ob3.getPemilik());

        pemilik = null;
        merk = null;
        warna = null;
        ob.hapus();
        ob = null;
        ob2 = null;
        ob3 = null;
    }
}
```

MODUL 6

LATIHAN SOAL

1. Buatlah aplikasi Data Mahasiswa sebagai berikut:

   a) Masukkan Nama Universitas! Buat static variable dan static method *setter getter*-nya!

   b) Masukkan data mahasiswa

      1) NIM

      2) NAMA

      3) ALAMAT

      4) JURUSAN : 61 = MATEMATIKA, 62 = BIOLOGI, 63 = KIMIA, 64 = FISIKA, 65 = TEKNIK INFORMATIKA, 66 = TEKNIK ARSITEKTUR

      Apakah Anda ingin memasukkan data lagi ? (Y) Ya ; (T) Tidak

      Jika *user* memasukkan “Y”, maka lanjut untuk memasukkan data mahasiswa lagi dan jika

      user memasukkan data “T”, maka aplikasi berhenti.

   Desainlah aplikasi Data Mahasiswa tersebut dengan konsep enkapsulasi, constructor dan inheritance (tentukan parent class dan child class). Setelah itu, implementasikan class-class yang telah didesain dengan membuat program sederhana yang memiliki fasilitas entri data Mahasiswa dan melihat daftar Mahasiswa yang telah dimasukkan.