Pertemuan 2.

Class, Objek, Methode dan Penerapan Parameter

A. Class

Class didefinisikan sebagai blue print atau prototype yang mendefinisikan variable-variabel dan metode-metode yang umum untuk semua objek. Classs berisi atribut dan perilaku obyek yang dibuatnya. Contohnya : class manusia, class tumbuhan, dll.

Bentuk umum :

➢ **Main Class/ Class Induk**

```
Modifier1 Class Namaclass modifier2 {
//tipedata namaatribut/variable
Public static void main (String[ ] args){
//character yang akan di tampilkan
 }
}
```

➢ **Sub Class/ Class Anak**

```
Modifier1 Class Namaclass modifier2 {
//tipedata namaatribut/variable
 }
```

Penamaan suatu class mempunyai aturan umum :

a. Tidak menggunakan spasi atau menggunakan spasi dengan diganti "_" (misal class Sarjana_Mipa)
b. Tidak menggunakan kata-kata yang telah dipakai oleh compiler (reserved Word, lihat bab sebelumnya)

Modifier saat deklarasi kelas adalah :

- Tidak ditentukan (default)
  Kelas tersebut dapat diakses oleh kelas lain dalam satu package.
- Public

Jika sebuah kelas tersebut dapat digunakan oleh kelas lain tanpa memperdulikan apakah kelas lain yang menggunkannya itu berasal dari package yang sama atau berbeda.

- Abstract
  Jika kelas tersebut memiliki abstract methode dan tidak dapat diinstansiasi menjadi sebuah objek.
- Final
  Kelas tidak akan dapat diturunkan lagi menjadi kelas turunan.

B. Data/ Atribut/ Variabel

Data adalah Komponen terkecil di dalam class atau karakteristik dari suatu class.

Contoh: karakteristik dari class manusia diantaranya; jenisKelamin, usia, warnaRambut, warnaKulit, dll.

Jenis type data

- Integer
  - keyword → byte, short, int, long
- Floating Point
  - keyword → float, double
- Character
  - char
- Boolean
- String

C. Object

Object adalah Bentuk nyata dari suatu class, dengan kata lain bentuk sederhana (instansiasi) dari class.

Bentuk umum:

NamaClass NamaObject =new NamaClass()

Contoh SubClass dan Class

1. Buat sub class berikut pd jcreator, simpan dengan nama contohjava.

contohjava.java

```
public class contohjava {
    String jeniskelamin="Laki-laki",
        warnarambut="Hitam",
        usia="27";
}
```

2. Kemudian buat untuk main class dan simpan dengan nama contohjavaberaksi

contohjavaberaksi.java *

```
public class contohjavaberaksi {
    public static void main(String[] args) {
        contohjava obj=new contohjava();
        System.out.println("Konsep OOP");
        System.out.println("Jenis Kelamin :"+obj.jeniskelamin);
        System.out.println("Warna Rambut :"+obj.warnarambut);
        System.out.println("Usia :"+obj.usia);
    }
}
```

3. Hasil programnya sbb:

General Output

```
Konsep OOP
Jenis Kelamin :Laki-laki
Warna Rambut :Hitam
Usia :27

Process completed.
```

D. Method

Sebuah method adalah bagian-bagian kode yang dapat dipanggil oleh program utama atau dari method lainnya untuk menjalankan fungsi yang spesifik di dalam kelas. Method dapat dibagi menjadi fungsi dan prosedur.

Fungsi adalah bagian atau sub dari program yang mempunyai algoritma tertentu dalam menyelesaikan suatu masalah dengan mengembalikan hasil.

Prosedur adalah bagian atau sub dari program yang mempunyai algoritma tertentu dalam menyelesaikan suatu masalah tanpa mengembalikan suatu nilai hasil. Secara umum method dalam java adalah sebuah fungsi.

Berikut adalah karakteristik dari method :

1. dapat mengembalikan satu nilai atau tidak sama sekali
2. dapat diterima beberapa parameter yang dibutuhkan atau tidak ada parameter sama sekali. Parameter bisa juga disebut sebagai argumen dari fungsi
3. setelah method telah selesai dieksekusi, dia akan kembali pada method yang Memanggilnya**.**

Deklarasi sebuah method

Method terdiri atas dua bagian yakni :

1. Method declaration

2. Method Body

Method dapat digambarkan sebagai sifat (behavior) dari suatu class. Untuk mendefinisikan method pada dalam class digunakan sintaks sintaks

[modifier] <tipe_data_return> **nama_method**([parameter]) { methode body }

Contoh : public int Perkalian (int y;int z) { methode body }

Modifier pada method

Modifier menentukan level pengaksesan sebuah method. Hal ini menentukan apakah sebuah method bias diakses oleh objek lain, objek anak, objek dalam satu paket atau tidak dapat diakses oleh suatu object sama sekali berikut adalah beberapa jenis level access:

- Public
  Atribut ini menunjukan bahwa fungsi/method dapat diakses oleh kelas lain.
- Private
  Atribut ini menunjukan bahwa fungsi atau method tidak dapat diakses oleh kelas lain
- Protected
  Atribut ini menunjukan bahwa fungsi atau method bisa diakses oleh kelas lain dalam satu paket dan hanya kelas lain yang merupakan subclass nya pada paket yang berbeda.

- Tanpa modifier
  Atribut ini menunjukan bahwa method dapat diakses oleh kelas lain dalam paket yang sama.

- Abstract
  Fungsi tidak memiliki implementasi.

- Final
  Method tersebut tidak dapat dioverride oleh kelas turunan.

- Static
  Method dapat diakses tanpa harus melakukan instantiasi terlebih dahulu.

Contoh Method

mobil.java

```
public class mobil {
/* class = metode + variabel-variabel */
 String warna,jenis;
 int tahunProduksi;

 void angkutan() {
    System.out.println("Mobil Termasuk Transportasi Darat");
    System.out.println("Jenis Mobil ini: " + jenis);
  }
}
```

mobilku.java *

```
public class mobilku {
  public  static  void  main(String[] args){
    /* Membuat object*/
     mobil  matic = new mobil();

     matic.warna = "Hitam";
     matic.jenis = " Mesin Otomatis";
     matic.tahunProduksi = 2006;
    /* memanggil metode */
     matic.angkutan();
     System.out.println("Warna: " + matic.warna);
     System.out.println("Tahun: " + matic.tahunProduksi);
  }
}
```

Hasilnya sbb:

```
General Output
Jenis Mobil ini:  Mesin Otomatis
Warna: Hitam
Tahun: 2006

Process completed.
```

E. Parameter

Bagian parameter diisi dengan parameter-parameter fungsi yang diperlukan. Parameter apabila lebih dari satu akan dipisahkan dengan tanda koma (“,”) parameter-parameter pada fungsi-fungsi di java akan di by pass value yang artinya pada tiap fungsi tidak akan dapat merubah isi dari variable parameter yang dimasukan.

Untuk memanggil method dapat digunakan sintaks sebagai berikut:

namaObyek**.**nama_method([parameter]);

Contoh :

sepeda.java

```
public class sepeda {
        int kecepatan, gir;

    // method dengan parameter
    void ubahGir(int pertambahanGir) {
       gir= gir+ pertambahanGir;
       System.out.println("Gir:" + gir);
    }

    void tambahKecepatan(int pertambahanKecepatan) {
       kecepatan = kecepatan+ pertambahanKecepatan;
       System.out.println("Kecepatan:" + kecepatan);
    }
}
```

sepedaBaru.java *

```
public class sepedaBaru {
          public static void main(String[] args) {
         // Membuat object
             sepeda sepedaku = new sepeda();

         /* memanggil atribut  dan memberi nilai */
             sepedaku.kecepatan=10;
             sepedaku.gir=2;

         // Memanggil method dan menunjuk nilai parameter
             sepedaku.tambahKecepatan(30);
             sepedaku.ubahGir(3);
     }
}
```

Hasilnya :

General Output

```
--------------------Configuration:
Kecepatan:40
Gir:5

Process completed.
```

Method dengan nilai balik - penggunaan fungsi dan prosedur

manusia.java

```
public class manusia {
   String nama,alamat,kontak;

   // Fungsi
   String ambilNama() {
   // Untuk mengembalikan nilai, menggunakan kata kunci return
   // Nilai balik di fungsi ini adalah nama dengan type String
   System.out.println("Nama : " + nama);
   return nama;
   }
     String ambilAlamat() {
     System.out.println("Alamat : " +alamat);
     return alamat;
    }
     // Prosedur
     void CetakKontak() {
     System.out.println("Kontak Aku : "+kontak);
     }
}
```

manusiaRamah.java

```
public class manusiaRamah {
  public static void main(String[] args) {
    manusia ramah = new manusia();
    ramah.nama = "Nurmalasari";
    ramah.alamat = "Kramat 25";
    ramah.kontak="nurmalasari.nmr@bsi.ac.id";

    // Mangambil nilai dari fungsi & methode
    String nama = ramah.ambilNama();
    String alamat = ramah.ambilAlamat();
    ramah.CetakKontak();
    }
}
```

General Output

```
--------------------Configuration: <Default>--------
Nama : Nurmalasari
Alamat : Kramat 25
Kontak Aku : nurmalasari.nmr@bsi.ac.id

Process completed.
```

Method Overloading

JAVA interpreter memiliki kemampuan untuk menggunakan beberapa cara pemanggilan method. Method overloading merupakan method dengan nama sama tetapi argument atau daftar parameter berbeda. Overloading method merupakan cirri-ciri dari pemograman berbasis obyek yang bersifat **polymorphisme**. Contohnya sbb :

anjing.java

```
public class anjing {
 void menggonggong() // tanpa parameter
  { System.out.println("guk guk"); }
 void menggonggong(String suaraGonggongan)
  { System.out.println(suaraGonggongan); }
 public static void main(String args[] )
  { anjing bleki = new anjing();
  anjing snoopy = new anjing();
  bleki.menggonggong();
  snoopy.menggonggong("wuf wuf");}
}
```

```
General Output
--------------------Configuration:
guk guk
wuf wuf

Process completed.
```

Method Static

Sebuah method static dapat diakses tanpa harus melakukan instantiasi terlebih dahulu. Pemanggilan method static dilakukan dengan format :

Nama_kelas.nama_method();

Nama_kelas diberikan bila method tersebut dipanggil dari kelas yang berbeda..

Contoh :

MyStatic.java

```
public class MyStatic {
    static void Luas() {
        int p = 10; int l = 2 ;
        System.out.println("Luas = "+ p*l );
        }
    public static void main(String[] args ) {
    Luas(); // pemanggilan method luas
    }
}
```

```
General Output
--------------------
Luas = 20

Process completed.
```