Lanjutan

CLASSES AND OBJECTS

:: Materi – 3 ::

Constructors

- ✓ Konstruktor menginisialisasi objek ketika objek tersebut dibuat.
- ✓ Konstruktor memiliki nama yang sama dengan kelasnya dan secara sintaksis mirip dengan suatu metode. Namun, konstruktor tidak memiliki tipe pengembalian yang eksplisit.

Constructors

- ✓ Semua kelas memiliki konstruktor, baik di definisikan atau tidak, karena Java secara otomatis menyediakan konstruktor secara default yang menginisialisasi semua variabel anggota ke nilai nol.
- ✓ Namun, setelah konstruktor ditentukan oleh programmer maka, konstruktor secara default tidak lagi digunakan.

Example


```java
public class data_mhs {

    int nim;
    String nama;

    //konstruktor tanpa parameter
    public data_mhs() {
    }

    //konstruktor menggunakan parameter
    public data_mhs(int nim, String nama) {
        this.nim = nim;
        this.nama = nama;
    }
}
```

Another Example


```
public class pencipta_java {

    //konstruktor menggunakan parameter
    public pencipta_java(String nama) {
        System.out.println("Nama : " +nama);
    }

    public static void main(String[] args) {
        pencipta_java mydata = new pencipta_java("James Gosling");
    }
}
```

Java Method

- ✓ Java method adalah kumpulan pernyataan yang dikelompokkan bersama untuk melakukan suatu operasi.
- ✓ Saat memanggil metode System.out. **println()**, sebenarnya sistem mengeksekusi beberapa pernyataan untuk menampilkan pesan di konsol.

Creating Method

- ✓ Java method adalah kumpulan pernyataan yang dikelompokkan bersama untuk melakukan suatu operasi.
- ✓ Saat memanggil metode System.out. **println()**, sebenarnya sistem mengeksekusi beberapa pernyataan untuk menampilkan pesan di konsol.

Example Java Method

Keterangan:

- ✓ **public static** – modifier
- ✓ **int** – return type
- ✓ **a, b** – formals parameter
- ✓ **methodName** – name of the method
- ✓ **int a, int b** – list parameter

```
public static int methodName(int a, int b) {
   //body
}
```

Definisi Metode

- ✓ Definisi metode terdiri dari header metode dan isi metode.
- ✓ Lihat Sintaksis berikut:

```
modifier returnType nameOfMethod (Parameter List) {
    // method body
}
```

Keterangan:

- ✓ **modifier** - mendefinisikan jenis akses dari metode dan bersifat opsional untuk digunakan.
- ✓ **returnType -** Metode dapat mengembalikan nilai.
- ✓ **nameOfMethod -** nama metode.
- ✓ **Parameter List** - Daftar parameter. Yaitu jenis, urutan dan jumlah parameter suatu metode. Ini bersifat opsional karena metode mungkin tidak berisi parameter apa pun.
- ✓ **method body** - Badan metode mendefinisikan apa yang dilakukan sesuai dengan pernyataan yang di berikan.

Example

Metode **minFunction()** mengambil dua parameter n1 dan n2 dan mengembalikan nilai minimum di antara keduanya.

```
public static int minFunction(int n1, int n2) {
    int min;
    if (n1 > n2) {
        min = n2;
    } else {
        min = n1;
    }

    return min;
}
```

Method Calling

Ada dua cara pemanggilan suatu metode yaitu:

- ✓ Metode yang mengembalikan suatu nilai
- ✓ Metode yang tidak mengembalikan nilai apa pun (no return value)

Ketika suatu program memanggil suatu metode maka, kontrol program akan ditransfer ke metode yang dipanggil. Metode yang dipanggil ini kemudian mengembalikan kontrol ke pemanggil.

Example

- ✓ Cara mendefinisikan suatu metode dan cara memanggilnya.
- ✓ Untuk metode yang di gunakan adalah program pada **slide hal.10**

```
public static void main(String[] args) {
    Scanner sc = new Scanner(System.in);
    int a;
    int b;

    System.out.print("Nilai a : ");
    a = sc.nextInt();
    System.out.print("Nilai b : ");
    b = sc.nextInt();

    int c = minFunction(a, b);
    System.out.println("Nilai Minimum = " + c);
}
```

Passing Parameters by Value

- ✓ *Passing Parameters by Value* maksudnya memanggil metode dengan parameter dan melalui metode ini, nilai argumen diteruskan ke parameter.
- ✓ Parameter dapat diteruskan berdasarkan nilai atau referensi.
- ✓ Contoh berikut ini, meneruskan parameter berdasarkan nilai dan nilai argumen tetap sama bahkan setelah pemanggilan metode.

Example


```
public class swappingExample {

    public static void main(String[] args) {
        int a = 30;
        int b = 45;
        System.out.println("Before swapping, a = " + a + " and b = " + b);

        // Invoke the swap method
        swapFunction(a, b);
        System.out.println("\nNow, Before and After swapping values will be same:");
        System.out.println("After swapping, a = " + a + " and b is " + b);
    }

    public static void swapFunction(int a, int b) {
        System.out.println("Before swapping(Inside), a = " + a + " b = " + b);

        // Swap n1 with n2
        int c = a;
        a = b;
        b = c;
        System.out.println("After swapping(Inside), a = " + a + " b = " + b);
    }
}
```

Method Overloading

- ✓ Ketika suatu kelas mempunyai jumlah metode dua atau lebih dengan nama yang sama tetapi parameternya berbeda maka, hal ini dikenal sebagai *method overloading*.
- ✓ Berbeda dengan *overriding*. Dalam *override*, suatu metode memiliki nama metode, jenis ataupun jumlah parameter yang sama.

NB: Untuk contoh dapat dicoba di praktikum modul3.

That’s all. Thank you very much! ☺

Any Questions?