MODUL

PEMOGRAMAN JAVA

Oleh:

Chalifa Chazar

Pertemuan 6

Kelas

Tujuan:

- **Mahasiswa mampu memahami konsep pemograman berorientasi object pada Java**
- **Mahasiswa mampu menggunakan konsep pemograman berorientasi pada pemograman Java**

Pustaka:

- **Bambang Hariyanto (2014): Esensi-Esensi Bahasa Pemograman Java (Revisi Keempat). Informatika. Bandung.**

Perkembangan gaya pemograman saat ini adalah pemograman berorientasi object. Pemograman berorientasi object memiliki banyak keunggulan dibandingkan dengan pemograman terstruktur, khususnya dalam menangani proyek yang kompleks. Pemograman menggunakan bahasa berorientasi object menawarkan fleksibilitas, penggunaan kembali (reuse able), dan kemudahan dalam perawatan.

Kelas merupakan konsep pokok di bahasa pemograman berorientasi object. Kelas merupakan bentukan logis yang menjadi landasan bangun seluruh bahasa orientasi object. Kelas mendefinisikan bentuk dan prilaku object. Kelas merupakan hal yang penting dalam java. Sembarangan konsep/abstraksi yang diimplementasikan di java harus dikapsulkan di dalam suatu kelas.

1. Pengertian Kelas

Pemograman Java tidak mungkin dipisahkan dari kelas. Pada awal pemograman, kelas hanya untuk mengkapsulkan metode (berbentuk fungsi main) untuk menunjukan sintaks dasar bahasa Java.

```
Public class FirstApp {
   public static void main(String[] args)
   ....
   }
}
```

Inti dari pemograman berorientasi object adalah pembentukan kelas. Kelas merupakan model atau abstraksi yang merepresentasikan object-object di dunia nyata. Berikut ini adalah beberapa pengertian kelas antara lain:

- Kelas adalah pendefinisian type baru.
- Kelas adalah template atau prototype yang mendefinisikan type object.
- Kelas merupakan sarana pengkapsulan kumpulan data dan metode-metode yang beroeperasi pada kumpulan data.
- Kelas adalah cetakan object (object harus merupakan instan suatu kelas).

Kelas digunakan untuk menciptakan banyak object. Object-object dari kelas yang sama mempunyai data sendiri tapi berbagi implementasi metode dengan object-object lain di satu kelas. Kelas dapat merupakan realisasi atau implementasi abstraksi di domain persoalan.

2. Deklarasi Kelas

Sintaks untuk mendeklarasikan kelas di Java adalah sebagai berikut.

```
Public class Manusia {
   public static void main(String[] args)
   ....
   }
}
```

2.1. Deklarasi Anggota Data

Sintaks untuk mendeklarasikan anggota data di Java adalah sebagai berikut.

```
Public class Manusia {
   public static void main(String[] args)
   String nama;
   String alamat;
   String telepon;
   }
}
```

2.2. Deklarasi Constructor

Format untuk pendeklarasian dan pendefinsian constructor adalah:

1. Nama contructor sama dengan nama kelas.

2. Sebelum itu dapat diberi access modifier untuk mengatur ketampakan constructor

Perhatikan contoh berikut ini. Kelas ke-1 mendeklarasikan kelas utama.

```
public class Manusia {

    public static void main(String[] args) {

        Mahasiswa maba = new Mahasiswa();
    }

}
```

Perhatikan contoh berikut ini. Kelas ke-2 mendeklarasikan constructor.

```
public class Mahasiswa {
   public Mahasiswa()
    {
        System.out.println("Constructor");
    }
}
```

2.3. Deklarasi Metode

Format umum pendeklarasian dan pendefinisian metode adalah:

1. Baris pertama adalah deklarasi metode, disebut **header metode** (method header)
2. Diikuti oleh **badan metode**, yang dimuat di dalam pasangan kurung kurawal

Perhatikan contoh deklarasi metode di bawah ini.

```
Header_Metode {
      Badan_Metode
}
```

Badan_Metode terdiri dari:

1. Deklarasi-deklarasi
2. Kalimat-kalimat

Badan metode membentuk satu block. Variable dapat dideklarasikan dimanapun di dalam block, block-block dapat bersarang.

Metode tidak dapat dideklarasikan di dalam metode yang lain.

Format dasar mendeklarasikan metode adalah sebagai berikut:

```
tipe-nilai-kiriman nama-metode(parameter1, parameter2, ..., parameterN) {
   deklarasi-deklarasi dan kalimat-kalimat
}
```

- » **nama-metode** harus nama identifier yang sah
- » **tipe-nilai-kiriman** adalah tipe yang dikirim ke pemanggil, tipe void mengindikasikan metode tidak mengirim nilai. Metode hanya mengirim paling banyal satu nilai.
- » **parameter-parameter** dideklarasikan sebagai deretan yang dipisah dengan koma di dalam pasangan kurung dalam hal ini masing-masing beisi tipe dan nama parameter. Jika metode tidak menerima satu argumenpun, daftar parameter kosong.

Terdapat tiga cara melepas kendali ke kalimat yang memanggil metode yaitu:

1. Jika metode tidak mengirim hasil, kendali lepas ketika aliran metode mencapai akhir metode yang ditandai dengan kurung kurawal tutup }, atau
2. Ketika kalimat

   ```
   return;
   ```

   dieksekusi.
3. Jika metode mengirim hasil, kalimat

   ```
   return ekspresi;
   ```

   Kemudian mengirim nilai yang dihasilkan ke pemanggil.

Perhatikan contoh pendeklarasian metode di bawah ini.

```
public class Matematika {
    public static int penjumlahan(int bilangan1, int bilangan2)
    {
        return bilangan1 + bilangan2;
    }
}
```

Perhatikan contoh pemanggilan metode penjumlahan di bawah ini.

```
public class latihan {

    public static void main(String[] args) {

        Matematika x = new Matematika();
        System.out.println(x.penjumlahan(1,2));

    }

}
```

3. Java Modifier

3.1. Pengelompokan Java Modifier

Modifier memberi dampak tertentu pada kelas,interface, metode dan variable. Java modifier terbagi menjadi kelompok berikut ini.

1. Modifier ketampakan (visibility modifier) disebut juga modifier pengaksesan (access modifier) berlaku untuk kelas, interface, metode dan variabel yaitu **default**, **public**, **protected**, **private**.
2. Final modifier berlaku untuk kelas, variabel dan metode, yaitu **final**.
3. Static modifier berlaku untuk variabel dan metode, yaitu **static**.
4. Abstract modifier berlaku untuk kelas dan metode, yaitu **abstract**.
5. Synchronized modifier berlaku metode, yaitu **synchronized**.
6. Native motifier berlaku untuk metode, yaitu **native**.
7. Strorage modifier berlaku untuk variabel, yaitu **transient**.

Access Modifier

No	Modifier	Kelas dan Interface	Metode dan Varibel
1	Defaul (tidak ada modifier)	Tampak di paketnya	Diwarisi oleh subkelasnya di paket yang sama dengan kelasnya. Dapat diakses oleh metode-metode di kelas-kelas yang sepaket.
2	Public	Tampak di manapun	Diwarisi oleh semua subkelasnya. Dapat diakses di manapun.
3	Protected	Tidak dapat diterapkan	Diwarisi oleh semua subkelasnya.

			Dapat diakses oleh metode-metode di kelas-kelas yang sepaket.
4	Private	Tidak dapat diterapkan	Tidak diwarisi oleh subkelasnya. Tidak dapat diakses oleh kelas lain.

No	Nama	Kata Kunci	Kelas	Interface	Metode	Variabel
1	Abstract modifier	Abstract	Kelas dapat berisi metode abstract. Kelas tidak dapat diinstaniasi Tidak mempunyai constructor	Optional untuk diberikan di interface karena interface secara inherenadalah abstract	Tidak ada badan metode yang didefinisikan. Metode memerlukan kelas kongkret yang merupakan subkelas yang akan mengimplementasikan metode abstract	Tidak dapat diterapkan
2	Final modifier	Final	Kelas menjadi tidak dapat digunakan untuk menurunkan kelas yang baru	Tidak dapat diterapkan	Metode tidak dapat ditimpa oleh metode di subkelas-subkelasnya	Berprilaku sebagai konstanta
3	Static modifier	Static	Tidak dapat diterapkan	Tidak dapat diterapkan	Mendefinisikan metode (milik) kelas. Dengan demikian tidak memerlukan instan object untuk menjalankannya. Metode ini tidak dapat menjalankan metode yang bukan static serta tidak dapat mengacu variabel yang bukan static	Mendefinisikan variabel (milik) kelas. Dengan demikian, tidak memerlukan instan object untuk mengacunya. Variabel ini dapat digunakan bersama oleh semua instan object
4	Synchronized modifier	Synchronized	Tidak dapat diterapkan	Tidak dapat diterapkan	Eksekusi dari metode adalah	Tidak dapat diterapkan

					secara mutual exclusive di antara semua thread. Hanya satu thread pada satu saat yang dapat menjalankan metode	pada deklarasi. Diterapkan pada instruksi untuk menjaga hanya satu thread yang mengacu variable pada satu saat
5	Native modifier	Native	Tidak dapat diterapkan	Tidak dapat diterapkan	Tidak ada badan yang diperlukan karena implementasi dilakukan dengan menggunakan bahasa lain misalnya C/C++	Tidak dapat diterapkan
6	Transient modifier	Transient	Tidak dapat diterapkan	Tidak dapat diterapkan	Tidak dapat diterapkan	Variable tidak akan diserialisasi
7	Volatile modifier	Volatile	Tidak dapat diterapkan	Tidak dapat diterapkan	Tidak dapat diterapkan	Variabel diubah secara asinkron. Kompilator tidak akan pernah melakukan optimasi atasnya

Latihan

1. Buat kelas matematika1 yang isinya mengandung 3 jenis metode untuk menghitung pengurangan, perkalian, dan pembagian. Untuk nilai diinput oleh user.
2. Buat kelas matematika2 yang isinya mengandung 3 jenis metode untuk menghitung luas bangun bujursangkar, persegi panjang, dan lingkaran. Untuk nilai diinput oleh user.

------- GOOD LUCK -------