Materi 4 : Enkapsulasi, Keywod This, Collection Type, dan Membuat CRUD Menggunakan Collection Type

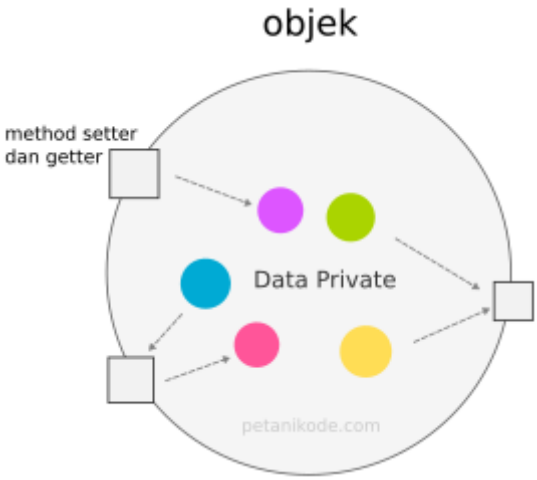
DAFTAR ISI

1.1 Enkapsulasi

Dalam OOP, enkapsulasi adalah cara menyembunyikan detail implementasi suatu kelas dari akses luar dan hanya mengekspos interface **`public`** yang dapat digunakan untuk berinteraksi dengan kelas tersebut.

Sederhananya, enkapsulasi (pembungkusan) adalah proses dimana data dibungkus dengan modifier **`private`** agar tidak bisa diakses secara langsung dari luar class.

Lalu bagaimana caranya agar data tersebut diakses dan bisa berguna? Caranya adalah dengan menggunakan method setter dan getter.



Mengapa harus dibuat seperti ini? Ada beberapa alasan kita harus menggunakan proses enkapsulasi:

- Untuk meningkatkan keamanan data
- Agar lebih mudah dalam mengontrol atribut atau method
- Class bisa dibuat menjadi read-only dan write-only
- Fleksibel, dalam artian developer dapat mengganti sebagian kode tanpa harus takut berdampak pada kode lain

1.1.1 Cara Membuat Method Setter & Getter

Untuk membuat method setter & getter sama saja seperti kita membuat method biasa. Misalnya kita memiliki sebuah data login user yang ingin kita rahasiakan.

Dalam contoh berikut terdapat atribut **`username`** dan **`password`** yang ingin kita rahasiakan datanya. Kemudian kita buat class dengan nama **`User`**, serta menambahkan kode berikut:

```
class User {
   // Data yang ingin dirahasiakan
   private String username;
   private String password;
```

```
    // method getter username
    public String getUsername() {
        return username;
    }

    // method setter username
    public void setUsername(String username) {
        this.username = username;
    }

    // method getter password
    public String getPassword() {
        return password;
    }

    // method setter password
    public void setPassword(String password) {
        this.password = password;
    }
}
```

Method setter dan getter harus diberikan modifier **`public`**, karena method ini akan diakses dari luar class. Perbedaan method setter dengan getter terletak pada nilai kembalian, parameter, dan isi method-nya.

Method setter tidak memiliki nilai kembalian **`void`** (kosong). Karena tugasnya hanya untuk mengisi data ke dalam atribut. Sedangkan method getter memiliki nilai kembalian sesuai dengan tipe data yang akan diambil.

1.1.2 Cara Menggunakan Method Setter & Getter

Cara menggunakannya pun cukup mudah, misalnya kita akses di method main. Kemudian kita ingin membuat program login seperti biasa. Maka berikut contoh programnya:

```
public static void main(String[] args) {
    User user = new User();
    Scanner input = new Scanner(System.in);

    System.out.print("Masukan username : ");
    String username = input.nextLine();
    user.setUsername(username);

    System.out.print("Masukan password : ");
    String pass = input.nextLine();
```

```
    user.setPassword(pass);

    if (user.getUsername().equalsIgnoreCase("admin") &&
        user.getPassword().equalsIgnoreCase("123")) {
        System.out.println("Login berhasil");
    } else {
        System.out.println("Login gagal");
    }
    input.close();
}
```

Program di atas akan menampilkan output "**`Login berhasil`**" jika kondisi **`true`**, dan "**`Login gagal`**" jika kondisi **`false`**.

1.2 This

Pada contoh sebelumnya, kita telah menerapkan proses enkapsulasi dengan method setter dan getter. Di dalam method tersebut kita menggunakan keyword **`this`** untuk mengisi variabel.

```
public void setUsername(String username) {
        this.username = username;
}
```

Jika keyword **`this`** di hapus, maka komputer akan bingung memilih variabel **`username`** yang dimaksud. Apakah variabel yang di class (**`private String username`**) ataukah variabel **`username`** yang ada di parameter **`setUsername`**.

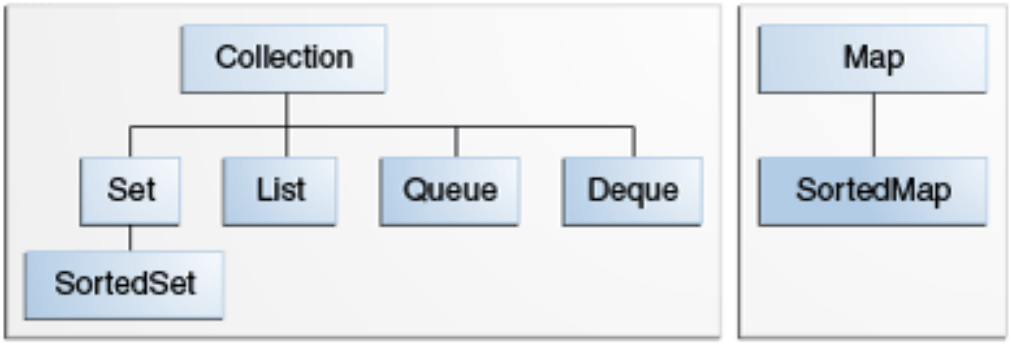
Jadi, keyword **`this`** digunakan sebagai referensi dari class itu sendiri.

1.3 Collection Type

Collection adalah kumpulan interface yang digunakan sebagai wadah untuk mengumpulkan beberapa elemen menjadi satu kesatuan. Apa itu interface? Interface sangat mirip dengan class pada umumnya, namun tidak memiliki atribut dan memiliki method yang di deklarasikan tanpa isi.

Jika definisi di atas terlalu sulit dimengerti, maka pengertian collection type secara sederhana adalah struktur data yang lebih kompleks dan lebih canggih dibandingkan dengan array.

Core collection interface dapat dirangkum pada gambar di bawah ini.



Dalam pelatihan ini, yang akan kita bahas adalah **`ArrayList`** yang salah satu dari interface **`List`**, dan **`HashMap`** sebagai salah satu implementasi dari interface **`Map`**.

1.3.1 ArrayList

List adalah suatu Collection di mana data yang masuk akan disimpan secara teratur. List mempunyai indeks berdasarkan urutan objek yang dimasukkan ke dalam list.

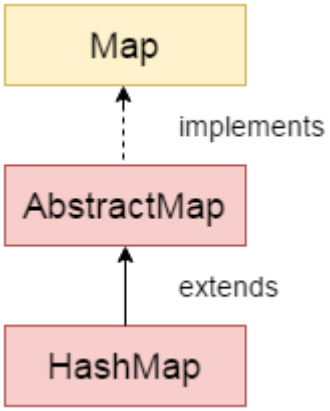
ArrayList merupakan salah satu contoh interface dari list yang mirip dengan Array. Perbedaannya hanya di ukurannya saja, ArrayList memiliki ukuran yang tidak tetap atau fleksibel sedangkan Array biasa ukurannya tidak bisa diubah.

Berikut adalah contoh penerapannya:

```
public static void main(String[] args) {
    ArrayList<String> list = new ArrayList<>();
    list.add("Freya");
    list.add("Zee");
    list.add("Wahyu");
    list.remove(2);
    list.set(1, "Shani");
    System.out.println(list);
}
```

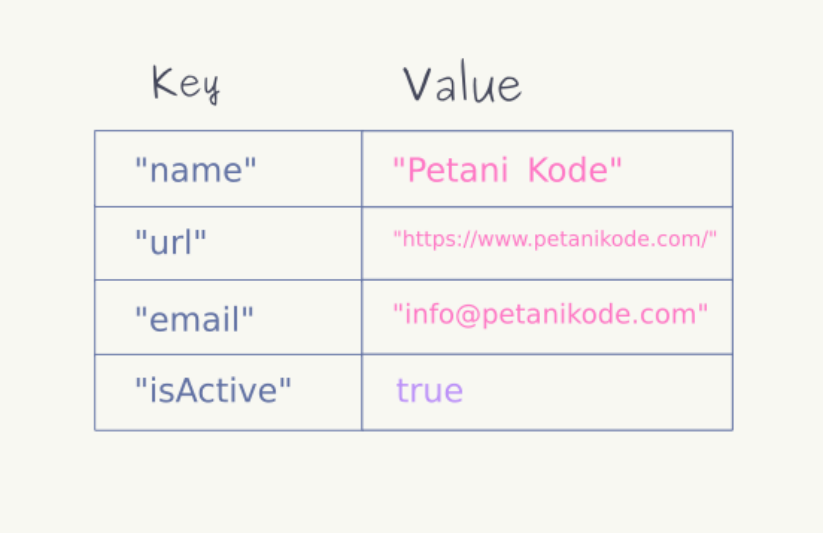
1.3.2 HashMap

Class HashMap merupakan class turunan dari class AbstractMap dan implementasi dari interface Map.



HashMap adalah sebuah class yang berisi sekumpulan pasangan nilai (value) dan kunci (key). Nilai bisa dalam bentuk string, integer, boolean, float, double, dan objek. Sedangkan untuk key biasanya dalam bentuk string dan integer.

Coba perhatikan gambar berikut ini.



Gambar di atas terdiri dari pasangan key dan value, seperti inilah isi dari class atau objek HashMap.

Berikut adalah contoh prgram menggunakan HashMap.

```
public static void main(String[] args) {
    HashMap<String, String> days = new HashMap<String, String>();

    // mengisi hashmap
    days.put("senin", "Menghapal vocab");
    days.put("selasa","Hapalan vocab");
    days.put("rabu","Ada tes vocab");
    days.put("kamis","Pengumuman nilai vocab");
    days.put("jumat","Kecewa dengan pengumuman vocab");
    days.put("sabtu","Dikasih vocab baru");
    days.put("minggu","Aku suka vocab :)");

    // cetak hashmap
    for (Map.Entry x : days.entrySet()) {
        System.out.println(x.getKey() + " : " + x.getValue());
    }
}
```

Kode di atas, menjelaskan jika kita ingin membuat sebuah HashMap yang berisi quotes of the days in PUB. Kemudian kita mencetaknya dengan menggunakan perulangan for each.

Bagaimana jika kita ingin mengganti value di HashMap, caranya adalah menggunakan method **`replace()`**. Berikut contohnya:

```
// mengganti value jumat
days.replace("jumat","Aku cinta PUB");
```

Bagaimana jika kita ingin menghapus value dari HashMap, caranya adalah menggunakan method **`remove()`**. Berikut cara penggunaanya:

```
// menghapus value minggu
days.remove("minggu");
```

Ada beberapa method lagi yang bisa kalian pakai dan pelajari, diantaranya adalah :

- **`clear()`**, untuk membersihkan isi HashMap
- **`isEmpty()`**, untuk mengecek apakah HashMap dalam keadaan kosong atau tidak
- **`size()`**, untuk mengambil ukuran HashMap
- **`values()`**, untuk mengambil semua value yang ada di HashMap
- **`keySet()`**, untuk mengambil semua key yang ada di HashMap
- **`clone()`**, untuk menggandakan objek HashMap