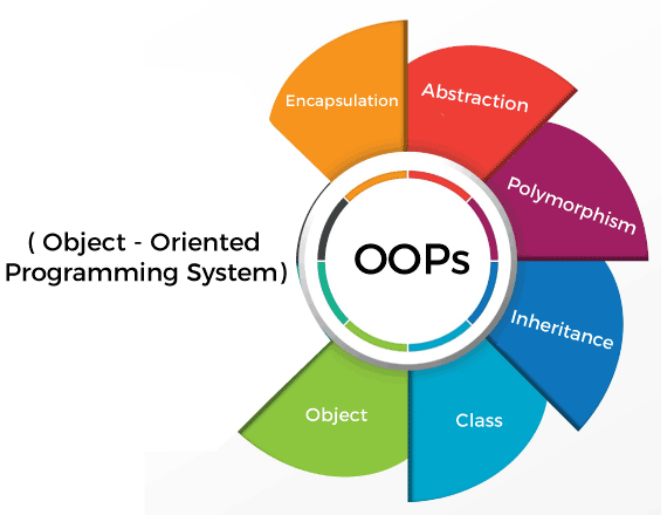
Materi 3 : Pengantar OOP, Class, Object, & Acces Modifier

DAFTAR ISI

1.1 Pengenalan OOP

OOP (Object Oriented Programming) atau dalam bahasa Indonesia dikenal dengan Pemrograman Berorientasikan Objek (PBO) merupakan sebuah paradigma atau teknik pemrograman yang berorientasi Objek.

Pada OOP, Fungsi dan variabel **dibungkus** dalam sebuah **objek** atau class yang dapat saling berinteraksi, sehingga membentuk sebuah program. Dengan demikian, tidak akan ada lagi kode yang berantakan.



1.2 Class

Dalam bahasa pemrograman Java, class adalah cetakan atau blueprint dari sebuah objek. Class berisi definisi variabel dan fungsi yang menggambarkan sebuah objek. Variabel di dalam class disebut sebagai **attribute** atau **properti**, sedangkan fungsi di dalam class disebut sebagai **method**.

```java
class Car {

    // attribute
    String nameCar;
    String brand;
    int price;

    // method
    void drive() {
        System.out.println("Mobil melaju...");
    }

    void stop() {
        System.out.println("Mobil berhenti.");
    }
}
```

Di dalam bahasa pemrograman java, terdapat beberapa jenis class yang bisa Anda pelajari sendiri. Diantaranya adalah sebagai berikut::

1. Reguler class
2. Abstract class
3. Interface
4. Enum class
5. Inner class
6. Lambda class
7. Anonymous class
8. Static class

1.3 Object

Dalam paradigma OOP, objek merupakan instance atau wujud nyata dari suatu kelas. Salah satu kegunaan objek adalah untuk mengakses berbagai fungsi dan metode pada kelas.

Karena objek merupakan instansiasi dari kelas, untuk menciptakan suatu objek kita perlu mendeklarasikan tiap objek yang dibuat dengan mengikutsertakan kelasnya. Pada Java untuk membuat sebuah objek terdapat 3 tahap :

- Deklarasi : mendeklarasikan nama sebuah objek
- Instansiasi : memerlukan sebuah perintah atau keyword `new` untuk menciptakan objek
- Inisialisasi : inisialisasi dari sebuah objek setelah perintah `new`

Agar lebih memudahkan lagi kita pakai contoh class bernama Car. Sekarang kita buat sebuah objek dari class Car tersebut, caranya adalah sebagai berikut:

```
public static void main(String[] args) {
    // deklarasi objek
    Car mobil;

    // instansiasi objek
    new Car();

    // inisialisasi objek
    mobil = new Car();

 }
```

Penjelasan dari kode di atas adalah :

- Kita membuat sebuah objek di method main (pembuatan objek bisa di method apa saja)
- Keyword `Car` artinya kita ingin mengakses sebuah class yang bernama `Car`
- Variabel `mobil` adalah nama dari objek yang sedang kita buat
- Keyword `new` berguna untuk menginstansiasi sebuah objek atau membuat objek baru

Dengan kode tersebut kita bisa dengan leluasa menggunakan properti maupun method yang berada di class Car. Misalnya kita ingin mengisi properti `nameCar` dengan `"Lamborghini"`. Maka kodenya adalah berikut ini.

```
public static void main(String[] args) {
   // instansiasi objek
   Car mobil = new Car();

   // mengisi value sebuah properti
   mobil.nameCar = "Lamborghini";

   // menampilkan isi sebuah properti
   System.out.println(mobil.nameCar);
}
```

1.4 Constructor

Constructor adalah method yang secara default sudah terbentuk ketika kelas dibuat. Ketika suatu kelas dibuat (instansiasi) maka konstruktor akan terpanggil juga. Constructor harus memiliki nama yang sama dengan nama kelasnya.

Intinya, kegunaan dari constructor ini adalah method khusus yang akan dipanggil pertama kali saat objek di buat. Misalnya kita gunakan class `Car`, kemudian kita buat sebuah constructor dengan kode berikut ini:

```
class Car {

   // membuat constructor
   Car(){
       System.out.println("Ini adalah constructor");
   }

   // statement lainnya...
}
```

Kemudian jika kita jalankan lagi programnya maka output "`Ini adalah constructor`" akan muncul di paling atas.

Karena constructor ini adalah sebuah method, maka kita bisa menambahkan parameter pada saat membuatnya. Perhatikan contoh berikut ini:

```
Car(String text){
        System.out.println(text + " buatan Indonesia");
}
```

Kemudian pada saat membuat objek (instansiasi) kita harus menambahkan sebuah argumen, berikut contoh pada saat pembuatan objek.

```
// instansiasi objek
Car mobil = new Car("Lamborghini");
```

Hal-hal yang harus diperhatikan dalam membuat constructor adalah :

- Nama constructor harus sama dengan nama class
- Tidak perlu memakai tipe data karena merupakan method khusus
- Constructor tidak memiliki return value

1.5 Access Modifier

Class dalam program Java dapat saling berhubungan dengan cara memberikan akses terhadap member mereka. Semua yang ada di dalam sebuah class (atribut/properti dan method) disebut **member**. Biasanya akan ada tingkatan akses yang disebut **modifier**.

Access modifier di dalam Object Oriented Programming (OOP) akan menentukan apakah kelas lain dapat menggunakan properti atau meng-invoke methods dari suatu kelas. Ada beberapa macam access modifier yang dapat digunakan yaitu `private`, `defaullt`, `protected`, dan `public`.

Modifier	Class yang sama	Package yang sama	Subclass	Global
`private`	✓			
`default`	✓	✓		
`protected`	✓	✓	✓	
`public`	✓	✓	✓	✓

1.5.1 Private

Access modifier `private` akan membatasi akses hanya di dalam class. `Private` biasanya digunakan sebagai modifier dari member dan metode suatu class. Berikut adalah contoh penggunaan modifier `private` adalah sebagai berikut.

```java
package com.pasim.application;

public class Car {
    private String nameCar;
    String brand;
    int price;
    private void drive() {
        System.out.println("Mobil melaju...");
    }
    void stop() {
        System.out.println("Mobil berhenti.");
    }
}

public class CarApplication {
    public static void main(String[] args) {
        Car mobil = new Car();
        System.out.println(mobil.drive()); // kode berikut akan error
    }
}
```

Dalam kode di atas, terdapat variabel **`nameCar`** yang memiliki modifier **`private`** dan method **`drive()`**. Jika kita mengakses kedua member tersebut di class lain seperti class **`CarApplication`** pada contoh di atas, maka hasilnya akan error.

1.5.2 Default

`Default` modifier berarti penulisan kodenya tanpa atribut modifier. Ini berlaku untuk semua kelas, member, atau fungsi yang kita tuliskan tanpa access modifier. Modifier **`default`** bisa diakses selama masih dalam satu package.

Dari contoh di atas member yang memiliki modifier adalah:

- Atribut **`brand`**
- Atribut **`price`**
- Method **`stop()`**

1.5.3 Protected

Access modifier **`protected`** bisa diakses selama masih dalam satu package. **`Protected`** memiliki sedikit perbedaan dengan **`default`** modifier. Perbedaannya adalah **`protected`** bisa diakses dari luar package. Akan tetapi, satu-satunya cara untuk akses dari luar package adalah kelas yang hendak mengakses, merupakan kelas turunannya.

Misalnya kita ubah modifier atribut brand dengan **`protected`**

```java
protected String brand;
```

dengan mengubah modifier menjadi `protected`, maka atribut tersebut bisa di akses oleh class `CarApplication`.

1.5.4 Public

Access modifier `public` bisa kita sebut sebagai modifier global. Artinya bisa diakses dari manapun bahkan package yang berbeda. Seperti pada contoh kode sebelumnya, class `Car` ditambahkan modifier public. Karena modifiernya `public` maka bisa diakses dari package lainnya.