



Java Object Oriented Programming

Praktikum Pemrograman Berorientasi Object 2024

Pengenalan Object Oriented Programming



Apa itu Object Oriented Programming?

- Object Oriented Programming adalah sudut pandang bahasa pemrograman yang berkonsep “objek”
- Ada banyak sudut pandang bahasa pemrograman, namun OOP adalah yang sangat populer saat ini.
- Ada beberapa istilah yang perlu dimengerti dalam OOP, yaitu: Object dan Class



Apa itu Object?

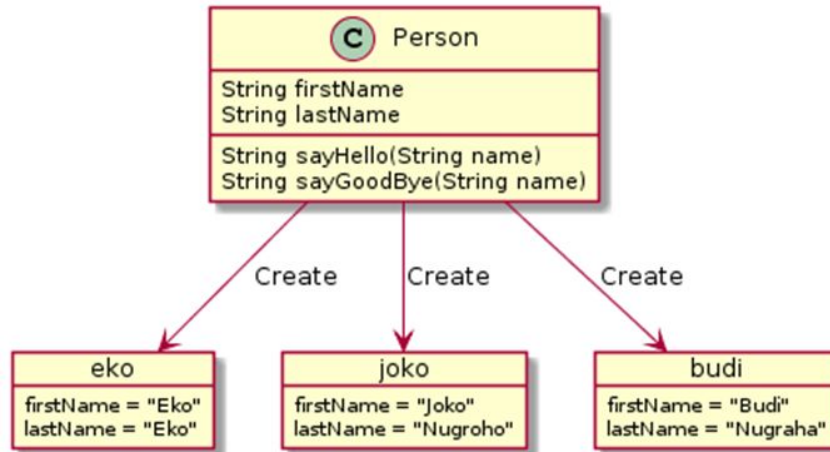
- Object adalah data yang berisi field / properties / attributes dan method / function / behavior
- Semua data bukan primitif di Java adalah object, dari mulai Integer, Boolean, Character, String dan yang lainnya



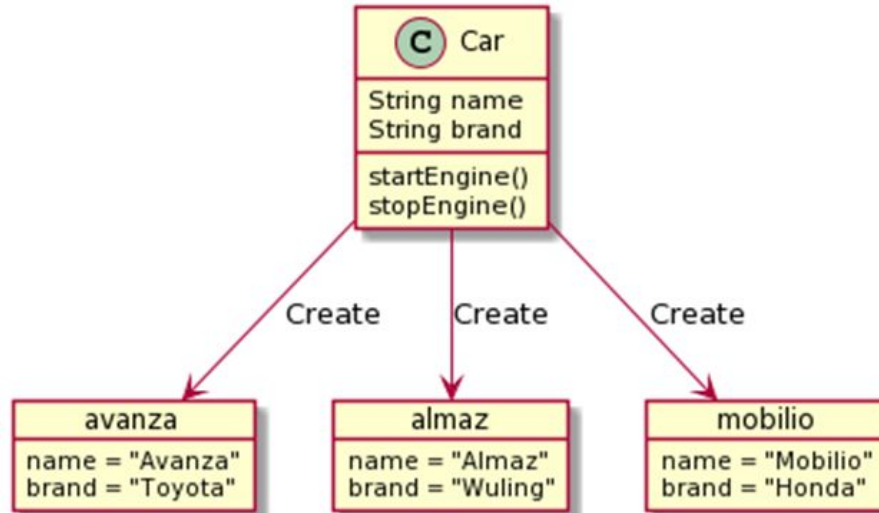
Apa itu Class?

- Class adalah blueprint, prototype atau cetakan untuk membuat Object
- Class berisikan deklarasi semua properties dan functions yang dimiliki oleh Object
- Setiap Object selalu dibuat dari Class
- Dan sebuah Class bisa membuat Object tanpa batas

Class dan Object : Person



Class dan Object : Car



Class

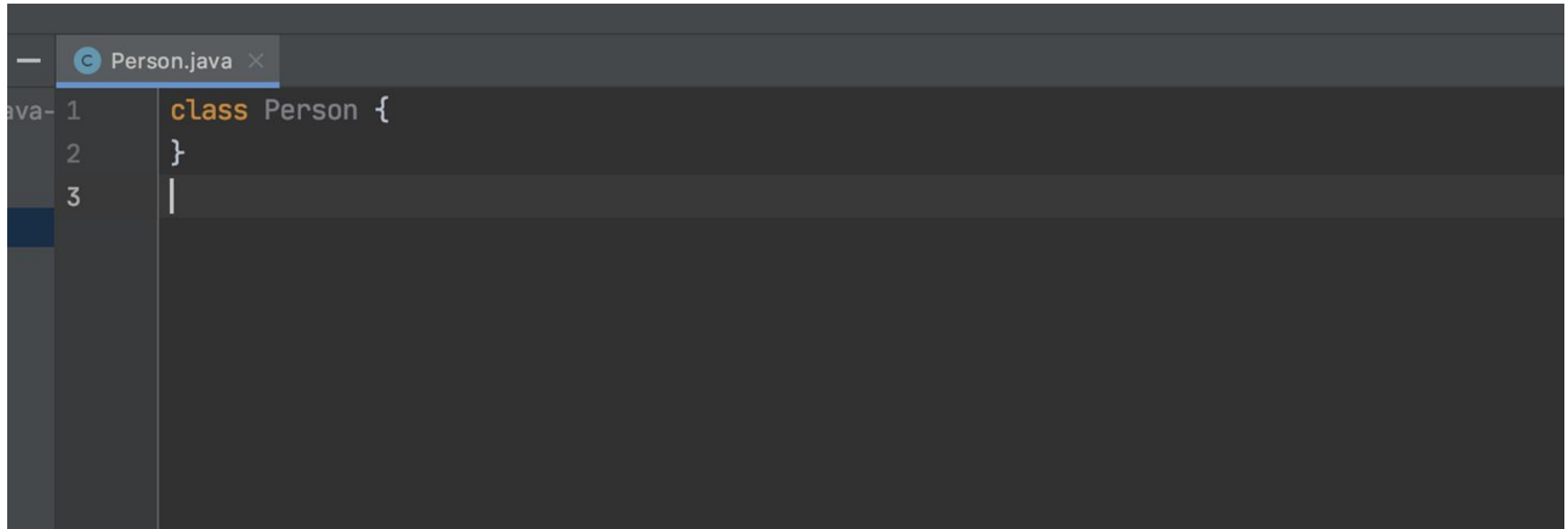


Membuat Class

- Untuk membuat class, kita bisa menggunakan kata kunci class
- Penamaan class biasa menggunakan format PascalCase



Kode : Class



```
— Person.java ×
ava- 1  class Person {
      2  }
      3  |
```

Object



Membuat Object

- Object adalah hasil instansiasi dari sebuah class
- Untuk membuat object kita bisa menggunakan kata kunci new, dan diikuti dengan nama Class dan kurung ()



Kode : Object

```
var person1 = new Person();  
Person person2 = new Person();  
  
Person person3;  
person3 = new Person();
```

```
- }
```

```
}
```

```
|
```

—

Field



Field

- Fields / Properties / Attributes adalah data yang bisa kita sisipkan di dalam Object
- Namun sebelum kita bisa memasukkan data di fields, kita harus mendeklarasikan data apa aja yang dimiliki object tersebut di dalam deklarasi class-nya
- Membuat field sama seperti membuat variable, namun ditempatkan di block class



Kode : Field

```
class Person1{  
    no usages  
    String name;  
    no usages  
    String address;  
    no usages  
    int age;  
    no usages  
    final String country = "Indonesia";  
}
```




Manipulasi Field

- Fields yang ada di object, bisa kita manipulasi. Tergantung final atau bukan.
- Jika final, berarti kita tidak bisa mengubah data field nya, namun jika tidak, kita bisa mengubah field nya
- Untuk memanipulasi data field, sama seperti cara pada variable
- Untuk mengakses field, kita butuh kata kunci . (titik) setelah nama object dan diikuti nama fields nya



Kode : Manipulasi Field

```
public class Before_Constructor {  
    public static void main(String[] args) throws Exception {  
        Person person1 = new Person();  
        person1.name = "Farhan";  
        person1.address = "Banyuwangi";  
        person1.age = 20;  
        //    person1.country = "Malaysia"; error karena tipe data final  
  
        System.out.println(person1.name);  
        System.out.println(person1.address);  
        System.out.println(person1.age);  
        System.out.println(person1.country);  
    }  
}
```

Method pada OOP



Method

- Selain menambahkan field, kita juga bisa menambahkan method ke object
- Cara dengan mendeklarasikan method tersebut di dalam block class
- Sama seperti method biasanya, kita juga bisa menambahkan return value, parameter dan method overloading di method yang ada di dalam block class
- Untuk mengakses method tersebut, kita bisa menggunakan tanda titik (.) dan diikuti dengan nama method nya. Sama seperti mengakses field



Kode : Method Pada Class Person(1)

```
void printAttribut(){  
    System.out.println(this.name);  
    System.out.println(this.address);  
    System.out.println(this.age);  
    System.out.println(this.country);  
}
```



Kode : Method (2)

```
public class Before_Constructor {  
    public static void main(String[] args) throws Exception {  
        Person person1 = new Person();  
        person1.name = "Farhan";  
        person1.address = "Banyuwangi";  
        person1.age = 20;  
        // person1.country = "Malaysia"; error karena tipe data final  
        person1.printAttribut();  
    }  
}
```

Constructor



Constructor

- Saat kita membuat Object, maka kita seperti memanggil sebuah method, karena kita menggunakan kurung ()
- Di dalam class Java, kita bisa membuat constructor, constructor adalah method yang akan dipanggil saat pertama kali Object dibuat.
- Mirip seperti di method, kita bisa memberi parameter pada constructor
- Nama constructor harus sama dengan nama class, dan tidak membutuhkan kata kunci void atau return value
- Saat kita membuat object baru, kita harus mengisi fieldnya satu persatu secara manual. Hal ini tentunya membuat ribet dan memakan waktu. Tujuan pembuatan constructor ini adalah untuk mempermudah pengisian field pada suatu object baru dibuat.



Kode : Membuat Constructor(1)

```
class Persons{  
    2 usages  
    String name;  
    2 usages  
    String address;  
    2 usages  
    int age;  
    no usages  
    final String country = "Indonesia";  
}
```



Kode : Membuat Constructor(2)

```
Persons(String name, String address, int age){  
    this.name = name;  
    this.address = address;  
    this.age = age;  
}  
  
1 usage  
void printAttribut() {  
    System.out.println(this.name);  
    System.out.println(this.address);  
    System.out.println(this.age);  
}
```



Kode : Membuat Constructor(3)

```
public class After_Constructor {  
    public static void main(String[] args) throws Exception {  
        Persons person1 = new Persons( name: "Farhan", address: "Banyuwangi", age: 21);  
        Persons person2 = new Persons( name: "Mulyono", address: "Bekerja", age: 10);  
        person1.printAttribut();  
        person2.printAttribut();  
    }  
}
```

Constructor Overloading



Constructor Overloading

- Sama seperti di method, di constructor pun kita bisa melakukan overloading
- Kita bisa membuat constructor lebih dari satu, dengan syarat tipe data parameter harus berbeda, atau jumlah parameter harus berbeda



Kode : Constructor Overloading

```
6  Person(String paramName, String paramAddress) {  
7      name = paramName;  
8      address = paramAddress;  
9  }  
10  
11  Person(String paramName) {  
12      name = paramName;  
13  }  
14  
15  Person() {  
16  
17  }
```



Kode : Menggunakan Constructor

```
3
4
5   var person1 = new Person( paramName: "Eko", paramAddress: "Subang");
6   var person2 = new Person( paramName: "Eko");
7   var person3 = new Person();
8
9   }
10 }
```



Memanggil Constructor Lain

- Constructor bisa memanggil constructor lain
- Hal ini memudahkan saat kita butuh menginisialisasi data dengan berbagai kemungkinan
- Cara untuk memanggil constructor lain, kita hanya perlu memanggilnya seperti memanggil method, namun dengan kata kunci `this`



Kode : Memanggil Constructor Lain

```
Person(String paramName, String paramAddress) {  
    name = paramName;  
    address = paramAddress;  
}  
  
Person(String paramName) {  
    this(paramName, paramAddress: null);  
}  
  
Person() {  
    this(paramName: null);  
}
```

Reference Pada Object



Referensi Pada Object

- Pada saat kita membuat object, kita harus tahu terlebih dahulu bagaimana perilaku object.
- Untuk mengetahui perilaku suatu object kita harus tahu terlebih dahulu reference atau alamat dari object itu.
- Hal ini diperlukan untuk memastikan keamanan data dari suatu object yang telah dibuat.
- Pada best praticenya, semua field pada kelas tidak boleh diakses mentah-mentah oleh kelas lain ataupun fungsi lain.
- Caranya adalah dengan mengasih access modifier kepada atribut kelas yang akan kita bahas setelah ini.



Kode: Class Buku

```
class Buku{  
    3 usages  
    String judul;  
    3 usages  
    String penulis;  
  
    1 usage  
    Buku(String judul, String penulis){  
        this.judul = judul;  
        this.penulis = penulis;  
    }  
  
    6 usages  
    void display(){  
        System.out.println("\nJudul\t: " + this.judul);  
        System.out.println("Penulis\t: " + this.penulis);  
    }  
}
```



Kode: Fungsi(2) Terletak di luar fungsi main

```
public static void fungsi(Buku dataBuku){  
    String addressDataBuku = Integer.toHexString(System.identityHashCode(dataBuku));  
    System.out.println("address dalam fungsi " + addressDataBuku);  
    dataBuku.penulis = "Haruki Mahalkami";  
}
```



Kode: Fungsi Utama(1)

```
Buku buku1 = new Buku( judul: "Killing Commandantore", penulis: "Haruki Murakami");  
buku1.display();  
  
// Menampilkan address  
String addressBuku1 = Integer.toHexString(System.identityHashCode(buku1));  
System.out.println(addressBuku1);  
  
// assignment object  
Buku buku2 = buku1;  
buku2.display();  
String addressBuku2 = Integer.toHexString(System.identityHashCode(buku2));  
System.out.println(addressBuku2);
```



Kode: Fungsi Utama(2)

```
// karena buku1 dan buku2 berada paddress atau referensi yang sama
buku2.judul = "Membunuh komandantur";
buku1.display();
buku2.display();

// kita akan memasukan object kedalam methods
fungsi(buku2);
buku1.display();
buku2.display();
```

Access Modifier



Access Modifiers

- Access modifier adalah kemampuan membuat class, field, method dan constructor dapat diakses dari mana saja
- Kali ini kita akan membahas access modifier untuk field dan method pada suatu class, dan di java ada setidaknya 4 access modifier, yaitu default, public, private, dan protected.
- Default dan Public memiliki sifat dapat ditulis dan dibaca secara umum, artinya kelas lain dapat mengaksesnya
- Private memiliki sifat hanya bisa dibaca dan ditulis pada kelas sendiri.
- Kali ini kita akan membahas public, private, dan default. Protected akan kita bahas setelah materi inheritance.



Kode: Kelas Player(1)

```
String name; // default, dia akan bisa dibaca dan ditulis dari luar class  
6 usages  
public int exp; // public, dia akan bisa dibaca dan ditulis dari luar class  
2 usages  
private int health; // private, hanya akan bisa dibaca dan ditulis di dalam class saja  
  
1 usage  
Player(String name, int exp, int health){  
    this.name = name;  
    this.exp = exp;  
    this.health = health;  
}
```



Kode: Player(2)

```
// default modifier access
2 usages
void display(){
    tambahExp(); // contoh mengakses private methods
    System.out.println("\nName\t: " + this.name);
    System.out.println("exp\t: " + this.exp);
    System.out.println("health\t: " + this.health); // membaca, tapi didalam class
}

// public
1 usage
public void ubahName(String nameBaru){
    this.name = nameBaru;
}

// private
1 usage
private void tambahExp(){
    this.exp += 100;
}
```



Kode: Fungsi main(1)

```
Player player1 = new Player( name: "Marni", exp: 0, health: 100);

// default
System.out.println(player1.name); // membaca data
player1.name = "Surti"; // Menulis data
System.out.println(player1.name); // membaca data

// public
System.out.println(player1.exp); // membaca data
player1.exp = 100; // Menulis data
System.out.println(player1.exp); // membaca data

// private (tidak bisa diakses)
// System.out.println(player1.health); // membaca data
// player1.health = 200; // Menulis data
// System.out.println(player1.health); // membaca data
```



Kode: Fungsi main(2)

```
// methods

// default
player1.display();

// public
player1.ubahName( nameBaru: "Tejo");
player1.display();

// private (tidak bisa diakses)
// player1.tambahExp();
```

Latihan



Latihan 1

Buatlah kelas dengan nama Stack yang memiliki field stck bertipe array berukuran 10 dan index bertipe integer. Buat fungsi push dan pop pada kelas tersebut! Kemudian pada main, buatlah 2 object stack yang masing-masing berisikan angka 0-9 dan 10-19. Tampilkan semua isi dari masing-masing object!



Latihan 2


Buatlah class yang memiliki field width, height, dan depth yang memiliki access modifier public dan kelas tersebut memiliki overload constructor. Yang pertama constructor yang memiliki 3 parameter, yaitu width, height, dan depth, yang kedua memiliki parameter length yang dimana constructor ini berarti width, height, dan depth memiliki nilai berupa length dan constructor yang tidak memiliki parameter yang berisikan semua fieldnya bernilai -1. Kelas tersebut memiliki fungsi volume. Aplikasikan ketiga constructor tersebut pada main dan tampilkan hasil volume pada terminal!



Latihan 3

Buatlah sebuah

1. Kelas Player yang memiliki field atau atribut name, health, level, weapon, dan armor yang memiliki access modifier public. atribut name bertipe data String, health bertipe data double, level bertipe data integer. weapon bertipe data object pada kelas Weapon, dan armor bertipe data object pada kelas Armor. Kelas player memiliki constructor, void attack, void defence, void equipWeapon, void equipArmor, dan void display.
2. Kelas Weapon yang memiliki field attackPower bertipe data double dan name bertipe data String serta memiliki access modifier public semua. Kelas Weapon memiliki constructor dan void display.
3. Kelas Armor memiliki field defencePower bertipe data double, name bertipe data String serta memiliki access modifier public semua. Kelas Armor memiliki constructor, dan void display.



Buatlah sebuah program game yang dimana player1 dapat menyerang player2 dan ketika player1 menyerang player2 maka terjadi 2 kemungkinan, jika armor player 2 lebih besar dari attack player 1, maka player 2 tidak terkena damage. Jika armor player 2 lebih kecil daripada attack player 2, maka attack player1 akan dikurangi armor player2 dan hasilnya merupakan attack yang terkena oleh player2 sehingga darah player2 berkurang sebanyak hasil pengurangan tersebut. Begitu pun sebaliknya. masing-masing player. Lakukan 3 kali penyerangan satu sama lain(Bebas).