
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?

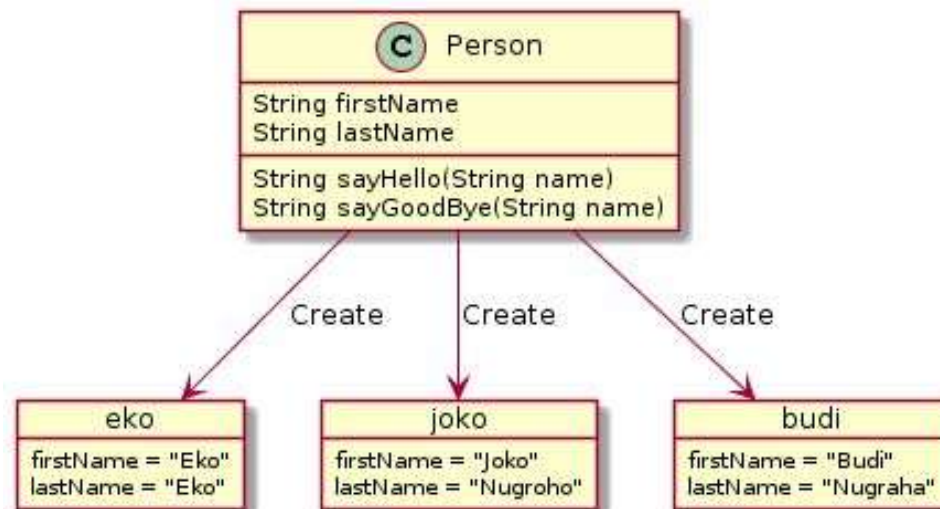
- Object adalah data yang berisi field / properties / attributes dan method / function / behavior
- Semua data bukan primitif di Java adalah object, dari mulai Integer, Boolean, Character, String dan yang lainnya



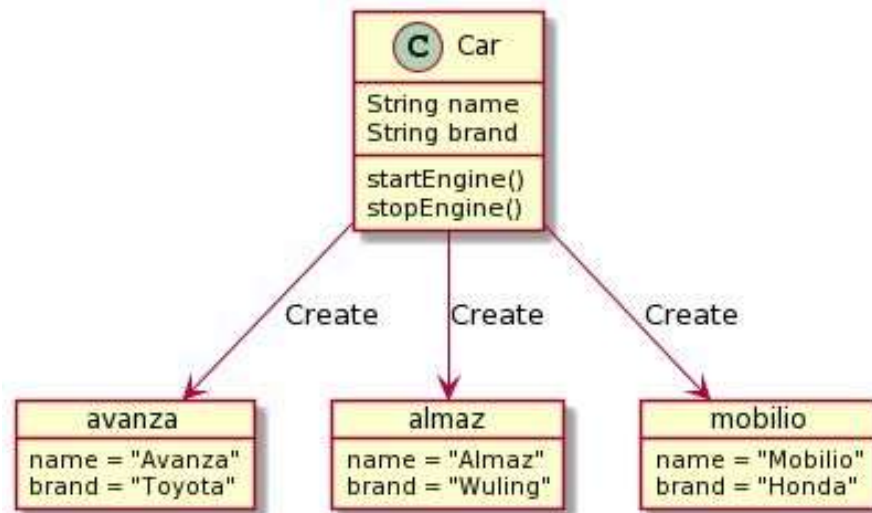
Apa itu Class?

- 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

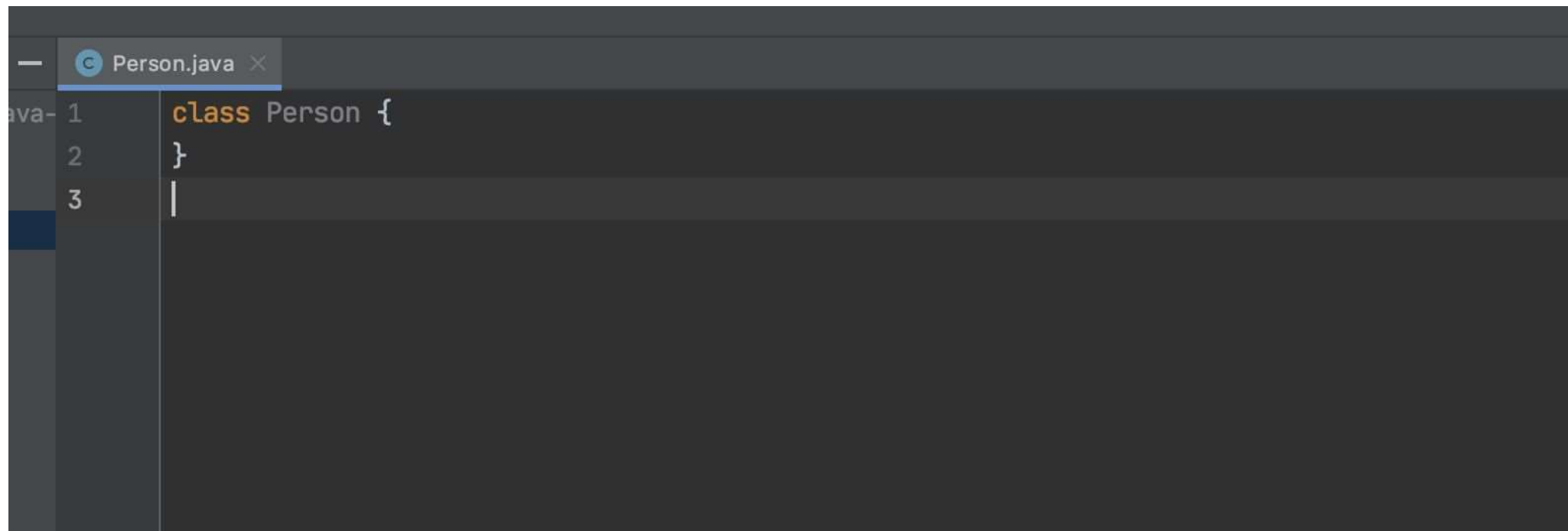


Membuat Class

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



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

```
Person.java x
1 class Person {
2     String name;
3     String address;
4     final String country = "Indonesia";
5 }
6
```



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

```
var person = new Person();  
person.name = "Eko Kurniawan";  
person.address = "Subang";  
// person.country = "Tidak Bisa Diubah";  
  
System.out.println(person.name);  
System.out.println(person.address);  
System.out.println(person.country);
```

```
}  
}
```

Method



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 (1)

```
class Person {  
    String name;  
    String address;  
    final String country = "Indonesia";  
  
    void sayHello(String paramName) {  
        System.out.println("Hello " + paramName + ", My Name is " + name);  
    }  
}
```



Kode : Method (2)

```
var person = new Person();  
person.name = "Eko Kurniawan";  
  
person.sayHello( paramName: "Budi");
```

```
}
```

```
}
```

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

Kode : Membuat Constructor

```
1  class Person {  
2      String name;  
3      String address;  
4      final String country = "Indonesia";  
5  
6      Person(String paramName, String paramAddress) {  
7          name = paramName;  
8          address = paramAddress;  
9      }  
10  
11     void sayHello(String paramName) {  
12         System.out.println("Hello " + paramName + ", My Name is " + name);  
    }
```




Kode : Menggunakan Constructor

```
3
4
5  var person = new Person( paramName: "Eko", paramAddress: "Subang");
6  person.name = "Eko Kurniawan";
7
8  person.sayHello( paramName: "Budi");
9
10 }
11 }
```

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 memanggилnya 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);  
}
```

Variable Shadowing



Variable Shadowing

- Variable shadowing adalah kejadian ketika kita membuat nama variable dengan nama yang sama di scope yang menutupi variable dengan nama yang sama di scope di atasnya
- Ini biasa terjadi seperti kita membuat nama parameter di method sama dengan nama field di class
- Saat terjadi variable shadowing, maka secara otomatis variable di scope di atasnya tidak bisa diakses

Kode : Variable Shadowing

```
final String country = "Indonesia";
```

```
Person(String name, String address) {  
    name = name; // field nama tidak berubah  
    address = address; // field address tidak berubah  
}
```

```
void sayHello(String name) {  
    System.out.println("Hello " + name + ", My Name is " + name); // field name tidak diakses  
}
```

```
void sayHello() {  
    this.sayHello(name: "Bos");
```

This Keyword



This Keyword

- Saat kita membuat kode di dalam block constructor atau method di dalam class, kita bisa menggunakan kata kunci this untuk mengakses object saat ini
- Misal kadang kita butuh mengakses sebuah field yang namanya sama dengan parameter method, hal ini tidak bisa dilakukan jika langsung menyebut nama field, kita bisa mengakses nama field tersebut dengan kata kunci this
- This juga tidak hanya digunakan untuk mengakses field milik object saat ini, namun juga bisa digunakan untuk mengakses method
- This bisa digunakan untuk mengatasi masalah variable shadowing

Kode : This Keyword

```
5
6 Person(String name, String address) {
7     this.name = name;
8     this.address = address;
9 }
10
11 void sayHello() {
12     this.sayHello(name: "Bos");
13 }
14
15 void sayHello(String name) {
16     System.out.println("Hello " + name + ", My Name is " + this.name);
17 }
```