KOSANKODING

OOP JAVA
PROGRAMMING WORKSHOP

By : KOSAN**KODING**

03 s/d 04 Maret 2018
- Kelas Diamond
(09.00 - 12.00 WIB)
- Kelas Gems
(13.00 - 16.00 WIB)

Kosan Koding
Perumahan Bumi Sadang Permai
Jl. Sadang/Cipadati, Blok C No 1, RT. 05/01
Cinunuk, Cileunyi, Bandung.

Scan with LINE App
to get Workshop
Location

FREE REGISTRATION

bit.ly/oop-workshop


KOSANKODING

Overview Day 1

1. Introduction
2. Kelas Dan Atribut
3. Objek Dan Memanggil Atribut
4. Method
5. Parameter
6. This
7. Overloading
8. Constructor
9. Package
10. Modifier
11. Keyword Static

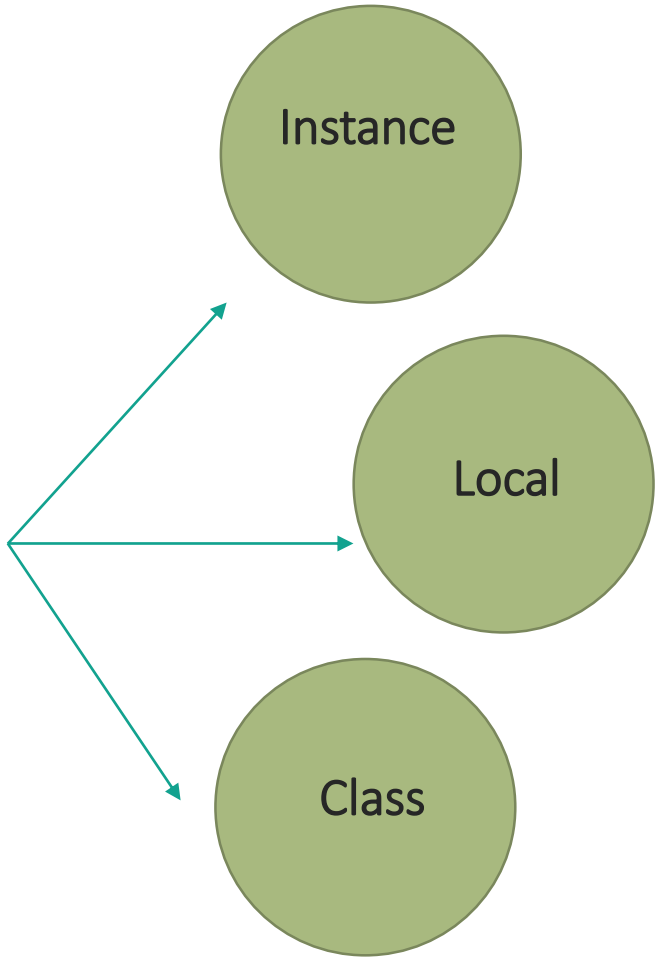
A.PENGANTAR

- ❑Apa itu OOP?
- ❑Tujuan OOP?
- ❑Harapan?

A.KELAS(CLASS)

= Rule

B. ATRIBUT(VARIABEL)



1.INSTANCE VARIABEL

Didalam class,diluar method

2.LOCAL VARIABEL

Didalam class,Didalam method

3.CLASS VARIABEL

Didalam class,diluar method dengan menggunakan static keyword

CONTOH CLASS & ATRIBUT

PENULISAN CLASS & ATRIBUT :

```
class MobilSaya{
    String merek;
    String warna;
    int harga;
    int nopolisi;
}
```

PENULISAN CLASS :

```
class MobilSaya{

}
```

OBJECT

- Apa itu OBJECT?

CLASS	→→→	OBJECT	→→→	ATRIBUT
TRANSPORTASI	→→→	MOBIL	→→→	Ban,Velg,Body,Spion
	→→→	Kambing	→→→	
	→→→		→→→	Besar,Belalai,Ekor,Gading

OBJECT:Aturan & Cara

1. Untuk membuat object pada java,bisa menggunakan keyword new
2. Untuk memanggil atribut bisa menggunakan tanda titik (.)
3. Semua atribut/variabel bisa dipanggil menggunakan object
4. Variabel/atribut,method yang dipanggil via object ,hanya yang dideklarasikan dalam class

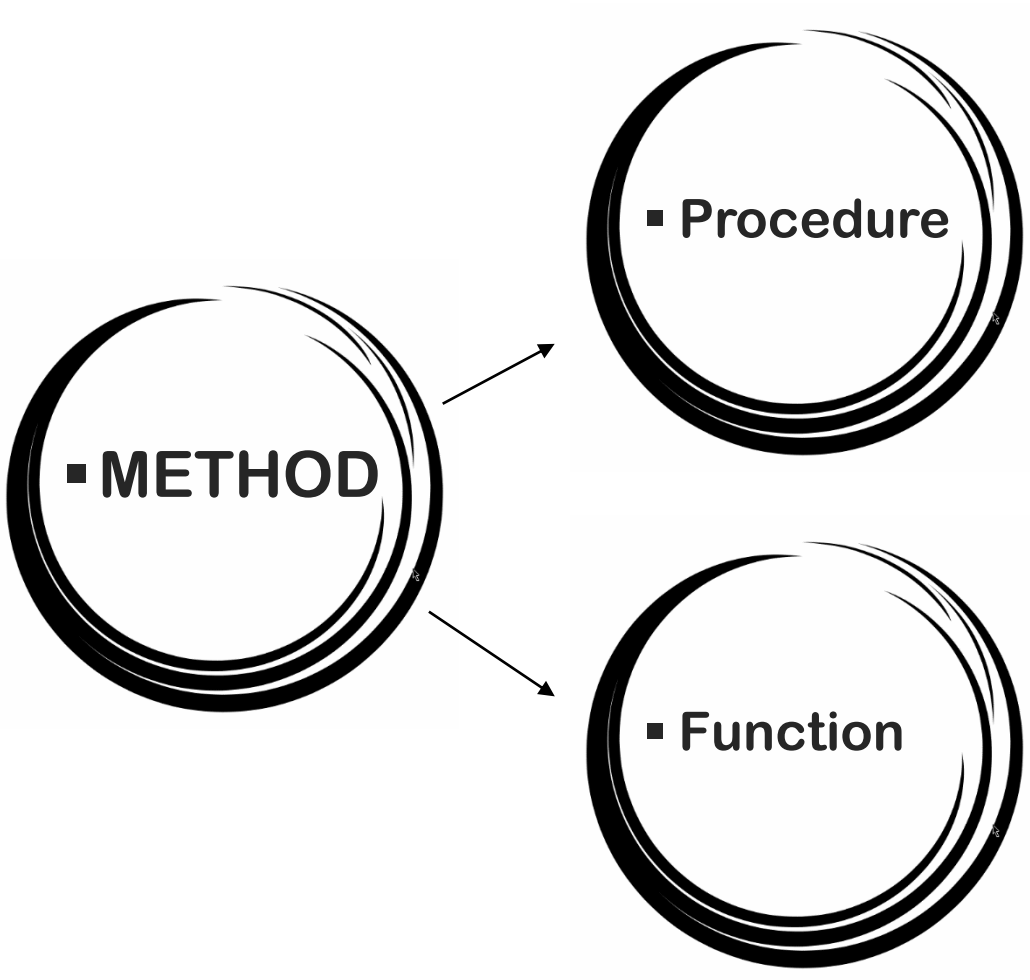
CARANYA:

1. NamaKelas objek = new NamaKelas;
2. objek.NamaAtribut;
3. objek.NamaAtribut=value;

Penulisan Object

```
Class MobilSaya{
    String merek;
    String warna;
    int harga;
    int nopolisi;
}
```

```
Class MobilSayaMain{
    MobilSaya objek = new MobilSaya();
    objek.merek="Toyota";
    objek.harga=650000;
}
```

Procedure	Function
Void	Tipe Data
Bisa mengembalikan banyak nilai(value)	Hanya bisa mengembalikan 1 nilai/value
Tidak harus mengembalikan value/nilai	Harus mengembalikan value/nilai = (return)

Contoh Procedure & Function

```
Class Mobil{
        void namaMethod(){
                //perintah
        }
}
```

```
Class MobilMain{
        public static void main (String[]args){
                Mobil objek = new Mobil();
                objek.namaMethod();
        }
}
```

```
Class Mobil{
        Int namaMethod{
                //Perintah
                return value(hasil);
        }
}
```

```
Class MobilMain{
        public static void main (String[]args){
                Mobil objek = new Mobil();
                objek.namaMethod();
        }
}
```

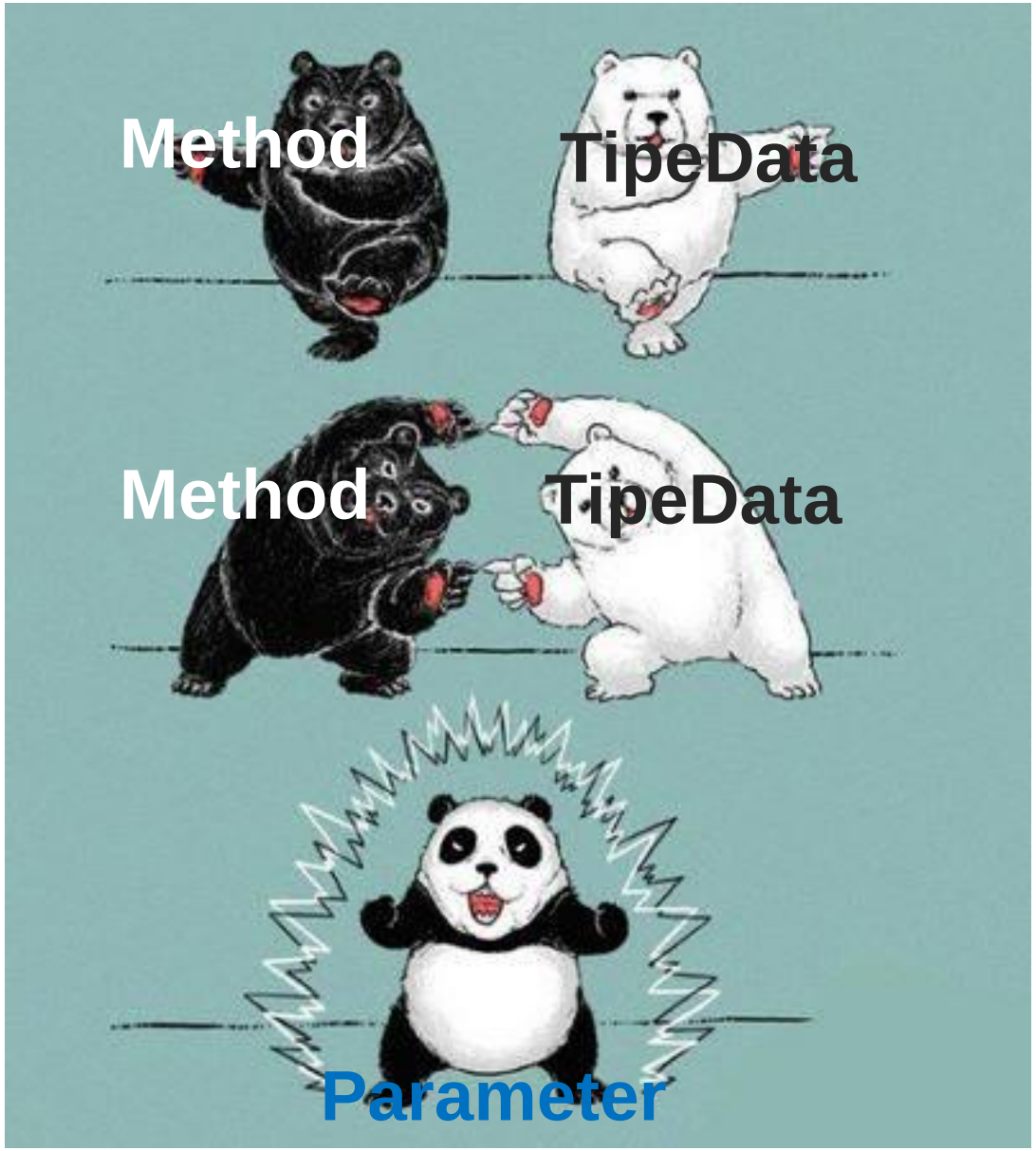
Parameter

Procedure:

```
Void namaProcedure(TipeData){
        //code
}
```

Function:

```
TipeData namaFunction(TipeData,TipeData){
        //code
}
```



Contoh Parameter:

```
int fungsikali(int angka1,int angka2) {
        int perkalian = angka1 * angka2;

        return perkalian;
}
```

```
//manggil objek
Parameter objek = new Parameter();

//manggil method
objek.fungsikali(3,5);
```

```
void prosedur(String nama){

        System.out.println("Nama Saya "+nama
}
```

```
//manggil objek
Parameter objek = new Parameter();

//manggil function
objek.prosedur("Karmila");
```

OUTPUT:
Procedur :
Function :

This

Apa Fungsi This?

```
Class namaKelas{
String nama;
void cariNama(String nama){
this.nama = nama;
    }
}
```


Overloading

Konsep Overloading

```
Void add (String nama){
//code
}
Void add (String nama, String alamat){
//code
}
Void add (String nama,String alamat, String kelas){
//code
}
```


```
Void add (int var,float var){
        //code
}

Void add (float var,int var){
        //code
}
```

Apakah ini Error?

```
Void add (float var, int var){
        //code
}

Void add (float tipe, int tipe){
        //code
}
```

Apakah ini Error?

Overloading dan Tipe Promosi

```
Void add (int var, Double tipe){
        //code A
}

Void add (float tipe, int var){
        //code B
}

//parameter
void Parameter (int tipe , float tipe);
```

Apa Output yang keluar ?

Constructor

Constructor Apakah Prosedur/Fungsi?

```
Class User{
User(){
//code
}

User (String nama){
//code
}
```


```
NamaKelas = new NamaConstructor();
NamaKelas = new NamaConstructor("Parameter");
```


package

```
Package namaPackage.subPackage.FileJava;

import namaPackage.subPackage.FileJava;

import namaPackage.subPackage.*;
```


Modifier

Class
Method
Variable

```
public static art{
}
public static int art;

public static void add()[
}
```


Default : Package ,class
Private : class
Public : All
Protected : package, subClass

Keyword static

```
Class DataMahasiswa {
Public static String nama;
Public static Method (){
    //code
}
}
//panggil : namaKelas.namaMethod();
```

Pengaruh Keyword static pada objek?

Overview Day 2

1. Encapsulation
2. Java Beans
3. Inner Clas
4. Inheritance
5. Overriding
6. Polimorphism
7. Konversi Kelas Polimorphism
8. Abstraction
9. Interface
10. Pewarisan Antar Interface

Encapsulation

Encapsulation adalah suatu proses membungkus sebuah atribut (variabel) ke dalam suatu method.

Di encapsulation variabel akan disembunyikan dari class dan package lain, dan hanya dapat di akses dan di rubah value atributnya bila menggunakan suatu method

Proses Encapsulation

1. Deklarasikan attributnya menggunakan hak akses private

2. buat 2 method(per attribut), untuk mengambil value dan untuk mengubah value dari attribut

Fungsi Encapsulation

- Bisa digunakan untuk melakukan sebuah perintah tertentu sebelum method dijalankan
- Di dalam method, kita dapat melakukan control apapun sebelum sebuah data disimpan
- Biasanya digunakan untuk melakukan validasi di dalam method tersebut
- Lebih secure, karena attributnya di bungkus ke dalam suatu method dan di method kita bisa lebih leluasa melakukan modifikasi perintah tertentu

Java Beans

Java beans merupakan implementasi dari encapsulation, bedanya hanya java beans sudah di set dalam pembuatan methodnya, menggunakan getNamaAtribut(untuk mengambil data) dan setNamaAtribut(untuk memodifikasi data) kecuali untuk boolean menggunakan isNamaAtribut(untuk ambil atribut) dan untuk memodifikasi atribut sama.

Inner Class

Inner Class merupakan kelas yang letak pembuatannya di dalam kelas

Cara pembuatan inner class sama seperti membuat class biasa, bedanya jika inner class dibuat di dalam suatu kelas yang sudah dibuat.

Inheritance

Inheritance atau pewarisan merupakan suatu proses menurunkan sifat (atribut, method) dari kelas utama ke suatu kelas lainnya, atau bisa dibilang kelas orang tua mewarisi atribut dan method ke suatu kelas anaknya

Setiap kelas orang tua dapat mewarisi lebih dari satu kelas anak. Namun, setiap kelas anak hanya dapat diwarisi dari satu kelas orang tua

Dijava untuk melakukan inheritance dapat menggunakan keyword extends

Overriding

Overriding merupakan proses mendeklarasikan ulang suatu method yang sudah ada di kelas orang tua, yang akan di baca ketika di panggil adalah method di kelas anak (setelah method di deklarasikan ulang), sedangkan method yang akan dipanggil di kelas oran tua, value nya akan berubah sesuai isi value deklarasi ulang di kelas anak

Keyword Super

- Keyword super digunakan untuk memanggil method di kelas orang tua

- Di deklarasikan di kelas anak fungsinya untuk menandakan bahwa method tersebut valuenya milik kelas orang tua

Polimorphism

Objek banyak bentuk, sebuah objek bisa memiliki banyak bentuk, bisa dari class Parent namun valuenya bisa dipanggil menggunakan class turunan

Parent Objek = new classTurunannya();

Konversi Kelas Polimorphism

Merupakan pengembalian objek ke kelas semula, sehingga kita bisa mengakses method yang ada di dalam objek polymorphismnya

```
KelasParent parrent = new KelasPoly();
KelasPoly poly = (KelasPoly) parent;
```


Abstraction

- Class dan Method bisa dibuat abstract
- Merupakan kelas yang tidak dapat digunakan objek kelasnya (tidak dapat di instantiated objeknya), kecuali dari kelas turunannya
- Jika ada sebuah method abstract pada maka semua methodnya perlu dideklarasikan di class objek pemanggilnya
- Pembuatan Method abstract harus di abstract class

```
public abstract class Mahasiswa{
    public abstract void namaMethod();
    public abstract void namaMethod2();
}
```

```
public Matakuliah extends Mahasiswa{
    public abstract void namaMethod(){
      //implementasi
    }
    public abstract void namaMethod2();{
      //implementasi
    }
}
```


Interface

- Bukan sebuah class, tapi bisa dibilang tipe class.
- Sebuah interface dapat mewarisi lebih dari satu interface ke suatu class.
- Semua method interface harus dideklarasilkan di class turunannya
- Interface dibuat menggunakan keyword interface
- Interface dapat diwariskan ke class menggunakan keyword implements

```
public interface Mahasiswa{
    public void namaMethod();
    void namaMethod2();
}
```

```
public class Matakuliah implements Mahasiswa, Dosen{
    public void namaMethod(){
      //implementasi
    }
    public void namaMethod2();{
      //implementasi
    }
}
```


Pewarisan Antar Interface

- Interface dapat diwariskan sesame interface lainnya
- Pewarisan antar interface menggunakan keyword extends

```
public interface Mahasiswa{
    public void namaMethod();
    void namaMethod2();
}

public interface Dosen{
    int NIK();
}
```

```
public interface Matakuliah extends Mahasiswa, Dosen{

}
```

Question?