UNIVERSITAS ISLAM INDONESIA

FUNDAMEN PENGEMBANGAN APLIKASI

Pertemuan 8

OOP: Kelas & Objek, Atribut, & Method

Tim Dosen FPA

UNIVERSITAS ISLAM INDONESIA

TOPIK MATERI

1. Kelas
2. Objek
3. Atribut
4. Method

UNIVERSITAS
ISLAM
INDONESIA

Kelas

Kelas
Definisi

UNIVERSITAS ISLAM INDONESIA

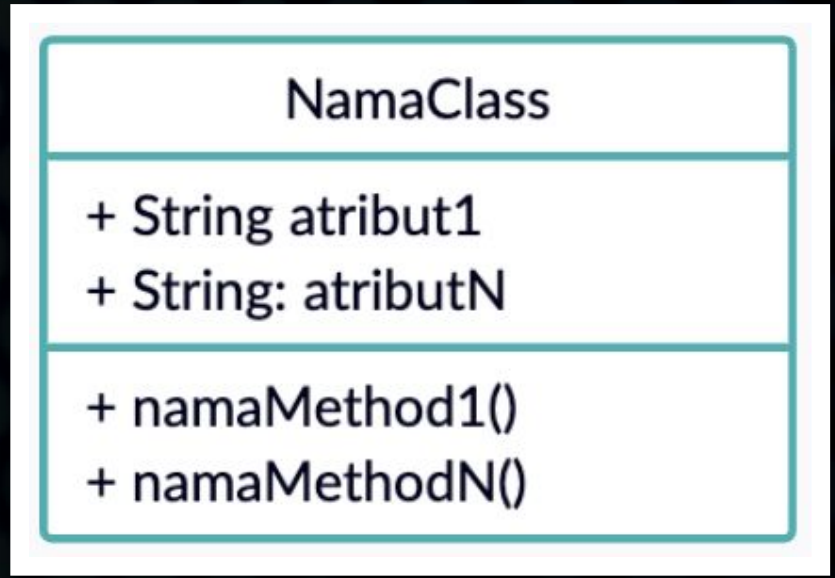
- Class adalah **rancangan** atau **cetak biru** dari sebuah objek.
- Kelas mendefinisikan:
  - **atribut** (biasa dalam pemrograman prosedural disebut ***variabel-variabel***) dan
  - **methods** (perilaku) umum dari sebuah objek tertentu.
- Deklarasi kelas menentukan ***apa saja yang dimiliki object (atribut/data)*** dan ***apa saja yang bisa dilakukan object (perilaku/method).***

Class
Definisi Kelas dalam Java

UNIVERSITAS ISLAM INDONESIA

```java
public class NamaClass {
    String atribut1;
    String atributN;

    void namaMethod1(){ ... }
    void namaMethodN(){ ... }
}
```



Kelas

Mendefinisikan Kelas

```
class NamaKelas {
    tipe atribut1;
    ….
    tipe atributN;

    tipe method1(daftar-parameter) {
    //kode untuk method1
    }

    ….

    tipe methodM(daftar-parameter) {
    //kode untuk methodM
   }
}
```

UNIVERSITAS ISLAM INDONESIA

Class Member

- Semua yang berada di dalam kelas (atribut dan method) disebut **member**
- Biasanya akan ada tingkatan akses yang disebut **modifier** (akan dibahas pada pertemuan Enkapsulasi)

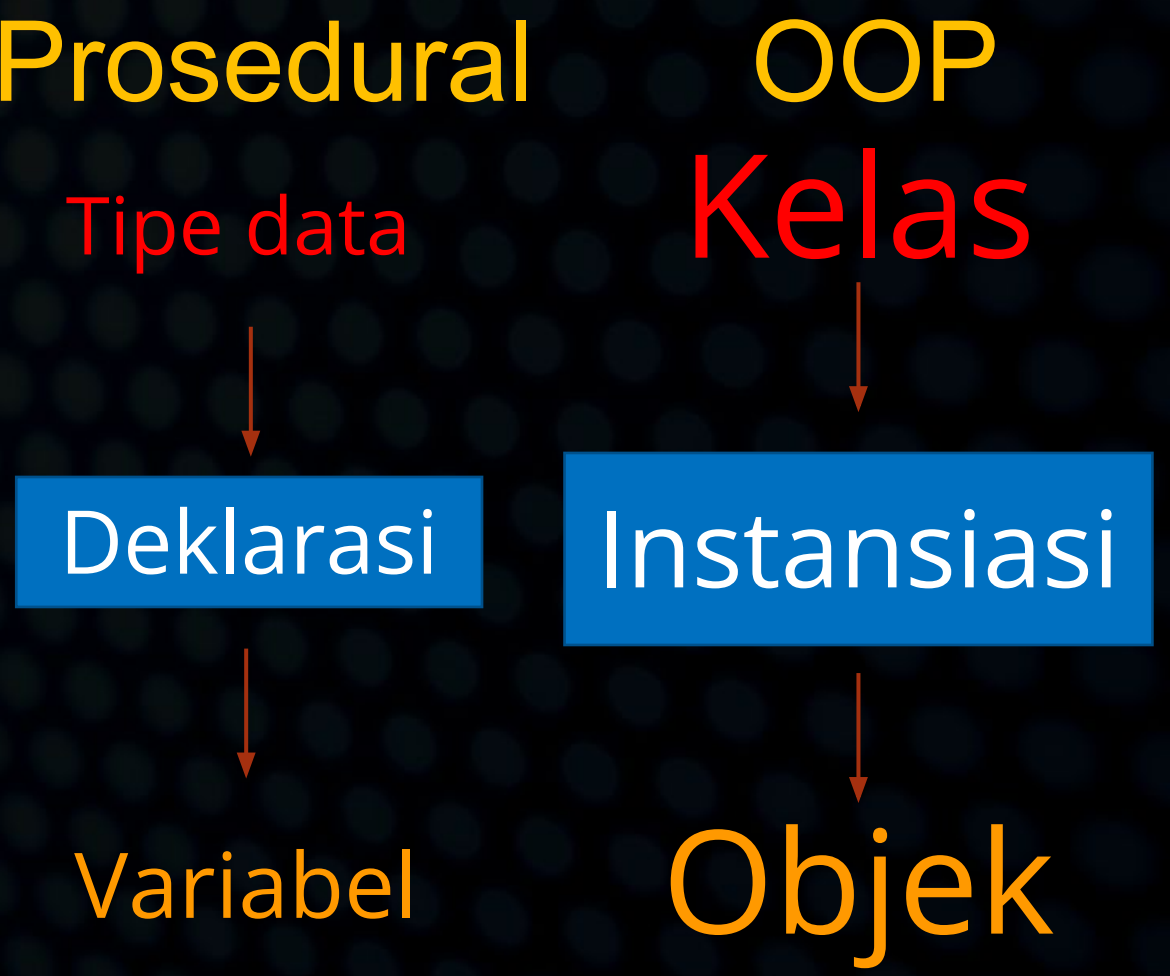
UNIVERSITAS
ISLAM
INDONESIA

Objek

Instansiasi

UNIVERSITAS ISLAM INDONESIA

- Dalam pemrograman prosedural, variabel dapat langsung digunakan setelah variabel dideklarasikan
- Dalam OOP, kelas harus **diinstansiasi** sebagai **objek** terlebih dahulu (tidak cukup hanya dengan deklarasi objek saja seperti di pemrograman prosedural)



UNIVERSITAS ISLAM INDONESIA

Kelas

Objek

Sebuah kelas **TIDAK BISA** digunakan untuk menangani sebuah tugas dalam program, sehingga harus dibuat terlebih dahulu **objek** dari kelas tersebut melalui proses **instansiasi**.

Objek
Definisi

UNIVERSITAS ISLAM INDONESIA

- Terdapat dua cara untuk menginstansiasi objek, cara (1) dan (2)
- Kata kunci **new** berfungsi untuk membuat objek baru dari suatu kelas

```
public class NamaClass {
    String atribut1;
    String atributN;

    void namaMethod1(){ ... }
    void namaMethodN(){ ... }
}
```

1
```
NamaClass namaObject = new NamaClass();
```

2
```
NamaClass namaObject;
namaObject=new NamaClass();
```

UNIVERSITAS ISLAM INDONESIA

Contoh

```
public class NamaClass {
    int x = 10;

  public static void main(String[] args) {
    NamaClass namaObject = new NamaClass();
    System.out.println(namaObject.x);
  }
}
```

```
run:
10
BUILD SUCCESSFUL (total time: 0 seconds)
```

- Membuat instansiasi objek dengan nama **namaObject** dari kelas **NamaKelas**
- Objek **namaObject** dapat mengakses atribut **x** yang dimiliki oleh **NamaClass**

UNIVERSITAS ISLAM INDONESIA

Multiple Objek

```
public class NamaClass {
  String nama = "Marjuki";
  int usia = 17;

  public static void main(String[] args) {
    NamaClass namaObject1 = new NamaClass();
    NamaClass namaObject2 = new NamaClass();
    System.out.println(namaObject1.nama);
    System.out.println(namaObject2.usia);
  }
}
```

```
run:
Marjuki
17
BUILD SUCCESSFUL (total time: 0 seconds)
```

- Instansiasi dua atau lebih objek dari kelas yang sama (**namaObject1**, **namaObject2**)
- Setiap objek dapat memanggil atribut yang sama atau berbeda yang dimiliki oleh kelas

UNIVERSITAS ISLAM INDONESIA

Multiple Kelas

```
public class NamaClass {
   int x = 11;
}
```

```
class ClassKedua {
  public static void main(String[] args) {
    NamaClass namaObject = new NamaClass();
    System.out.println(namaObject.x);
  }
}
```

```
run:
11
BUILD SUCCESSFUL (total time: 0 seconds)
```

- Untuk mengakses atribut yang berada pada kelas lain, perlu untuk menginstansiasi kelas tersebut menjadi objek terlebih dulu
- Membuat objek **namaObjek** yang di-instansiasi kelas **NamaClass**

UNIVERSITAS
ISLAM
INDONESIA

Atribut

UNIVERSITAS ISLAM INDONESIA

Mengakses Atribut

- Setelah membuat objek, kita bisa mengakses atribut dan method dari objek tersebut.
- Tanda titik (.) berfungsi untuk mengakses atribut dan method dari sebuah objek

```
namaObject.namaMethod1();
namaObject.atribut1;
```

<nama objek> . <nama method>
<nama objek> . <nama atribut>

UNIVERSITAS ISLAM INDONESIA

Mengakses Atribut

```java
public class NamaClass {
  int x = 7;

  public static void main(String[] args) {
    NamaClass namaObject = new NamaClass();
    System.out.println(namaObject.x);
  }
}
```

```
run:
7
BUILD SUCCESSFUL (total time: 0 seconds)
```

```java
public class NamaClass {
   String x = "Berlatih";

  public static void main(String[] args) {
    NamaClass namaObject = new NamaClass();
    System.out.println(namaObject.x);
  }
}
```

```
Berlatih
BUILD SUCCESSFUL (total time: 0 seconds)
```

- Kita dapat mengakses atribut suatu kelas setelah terlebih dulu menginstansiasi kelas tersebut menjadi objek (**namaObject**).
- **namaObject.x** → objek **namaObject** mengakses atribut **x**

Update Nilai Atribut

UNIVERSITAS ISLAM INDONESIA

```java
public class NamaClass {
  String x;

  public static void main(String[] args) {
    NamaClass namaObject = new NamaClass();
    namaObject.x = "Terus Berlatih";
    System.out.println(namaObject.x);
  }
}
```

```
run:
Terus Berlatih
BUILD SUCCESSFUL (total time: 0 seconds)
```

```java
public class NamaClass {
  String x = "Terus Berlatih";

  public static void main(String[] args) {
    NamaClass namaObject = new NamaClass();
    namaObject.x = "Terus Berlatih dan Pantang Menyerah";
    System.out.println(namaObject.x);
  }
}
```

```
run:
Terus Berlatih dan Pantang Menyerah
BUILD SUCCESSFUL (total time: 0 seconds)
```

- Bagian kiri, menambahkan nilai ke dalam atribut **x** (**namaObject.x=...**)
- Bagian kanan, update nilai atribut **x** (**namaObject.x=...**)

UNIVERSITAS ISLAM INDONESIA

Multiple Objek

```
public class NamaClass {
  int x = 1;

  public static void main(String[] args) {
    NamaClass namaObject1 = new NamaClass();
    NamaClass namaObject2 = new NamaClass();
    namaObject2.x = 2;
    System.out.println(namaObject1.x);
    System.out.println(namaObject2.x);
  }
}
```

```
run:
1
2
BUILD SUCCESSFUL (total time: 0 seconds)
```

Jika terdapat objek-objek yang merujuk ke kelas yang sama, kita dapat mengubah nilai atribut tanpa mengubah nilai atribut pada objek lainnya

UNIVERSITAS ISLAM INDONESIA

Multiple Atribut

```
public class NamaClass {
  String namaDepan = "Marco";
  String namaBelakang = "Van Basten";
  int usia = 24;

  public static void main(String[] args) {
    NamaClass namaObject = new NamaClass();
    System.out.println("Nama saya : " + namaObject.namaDepan + " " + namaObject.namaBelakang);
    System.out.println("Usia saya: " + namaObject.usia);
  }
}
```

```
run:
Nama saya : Marco Van Basten
Usia saya: 24
BUILD SUCCESSFUL (total time: 0 seconds)
```

Satu objek dapat mengakses satu atau beberapa atribut sekaligus

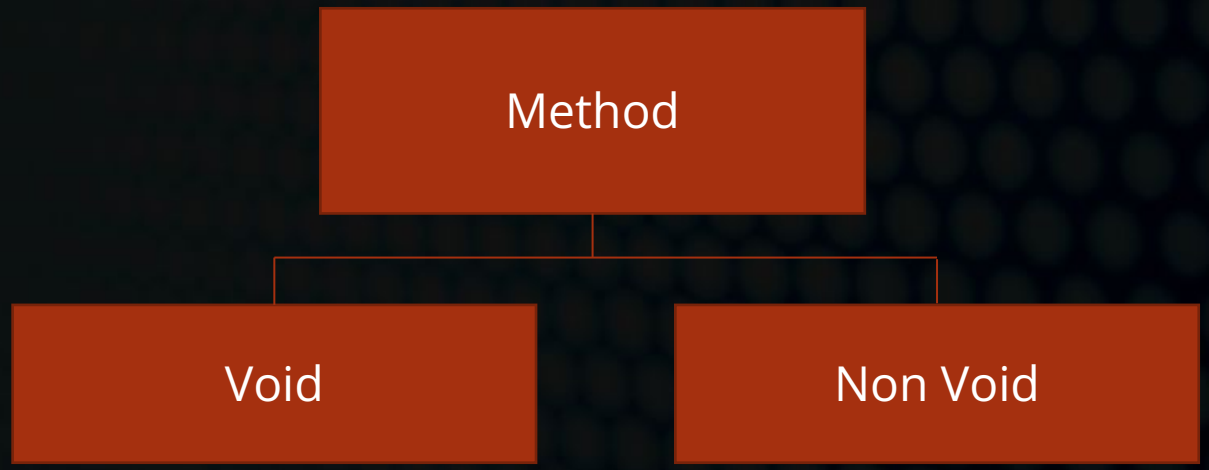
UNIVERSITAS
ISLAM
INDONESIA

Method

Method
Definisi

UNIVERSITAS ISLAM INDONESIA

- **Method** menentukan apa saja yang bisa dilakukan objek dari kelas tertentu



Tanpa *return value*

Dengan *return* value

* Sudah dipelajari pada pertemuan sebelumnya

UNIVERSITAS ISLAM INDONESIA

Mengakses Method

```
public class KendaraanRoda4 {
    public String warnaToString(String kendaraan, String warna){
      return "Warna " + kendaraan + " ini adalah " + warna;
    }

    public String kecepatanToString(String kendaraan, int maxKecepatan){
      return "Kecepatan tertinggi " + kendaraan + " ini adalah: " + maxKecepatan;
    }

     public static void main(String[] args) {
       KendaraanRoda4 mobil = new KendaraanRoda4();
       System.out.println(mobil.warnaToString("mobil", "merah"));
       System.out.println(mobil.kecepatanToString("mobil", 180));

     }
}
```

```
run:
Warna mobil ini adalah merah
Kecepatan tertinggi mobil ini adalah: 180
BUILD SUCCESSFUL (total time: 0 seconds)
```

- Pertama, kita membuat objek **mobil** dari kelas **KendaraanRoda4**
- Selanjutnya, objek **mobil** memanggil method **warnaToString** dan **kecepatanToString**
- Method **warnaToString** dengan 2 parameter bertipe sama, sedangkan **kecepatanToString** memiliki 2 parameter dengan tipe berlainan

Konstruktor
Definisi

UNIVERSITAS ISLAM INDONESIA

- Konstruktor adalah method khusus yang didefinisikan di dalam kelas dan akan **dipanggil secara otomatis** tiap kali terjadi instansiasi objek.
- Konstruktor itu sendiri berfungsi untuk melakukan **inisialisasi nilai** terhadap data-data yang terdapat pada kelas yang bersangkutan.
- Jika kita tidak mendefinisikan konstruktor pada kelas yang kita buat, secara otomatis Java akan membuatkannya untuk kita. Konstruktor semacam ini dinamakan dengan **default constructor**.
- Sama halnya seperti method, konstruktor juga dapat memiliki parameter dan juga dapat **di-overload*) (didefinisikan dalam beberapa versi tetapi dengan satu nama saja, setiap versi mempunyai parameter yang berbeda)**.

*)Melakukan overload berarti mendeklarasikan lebih dari 1 (satu) method dengan kesamaan nama namun parameter yang berbeda, akan dibahas secara tersendiri

Konstruktor

```java
public class User {
    public String username;
    public String password;

    public User(String username, String password){
        this.username = username;
        this.password = password;
    }
}
```

- Nama konstruktor **HARUS SAMA** dengan nama kelas

UNIVERSITAS ISLAM INDONESIA

Kata Kunci **This**

- Kata kunci **this** digunakan untuk mewakili kelas yang bersangkutan, biasanya digunakan pada blok *statement* ***method*** untuk **mengacu pada objek** yang memiliki method tersebut.

- Kata kunci **this** diperlukan terutama pada method dengan **nama parameter yang sama dengan atribut** yang dimiliki kelas yang bersangkutan.

This (contoh)

```
public class NamaClass {
    String nim;
    String nama;


    NamaClass (String nim, String nama){
      this.nim = nim;
      this.nama = nama;
    }

    public String toString(){
      return (nim + " " + nama);
    }

    public static void main(String[] args) {
      NamaClass mahasiswa1 =
              new NamaClass("20523000", "Juku");
      NamaClass mahasiswa2 =
              new NamaClass("20523001", "Eja");

      System.out.println(mahasiswa1.toString());
      System.out.println(mahasiswa2.toString());
    }
}
```

```
run:
20523000 Juku
20523001 Eja
BUILD SUCCESSFUL (total time: 0 seconds)
```

- Kelas **NamaClass** memiliki atribut yang sama dengan dimiliki oleh method **NamaClass** (nim, nama)
- **This** digunakan untuk merujuk objek yang diinstansiasi dari **NamaClass** (dalam contoh ini adalah objek yang mengakses konstruktor)

(Tanpa) This

```
public class Mahasiswa {
    String nim;
    String nama;


    Mahasiswa (String nim, String nama){
      nim = nim;
      nama = nama;
    }

    public String toString(){
      return (nim + " " + nama);
    }

    public static void main(String[] args) {
      Mahasiswa mahasiswa1 =
              new Mahasiswa("20523000", "Juku");
      Mahasiswa mahasiswa2 =
              new Mahasiswa("20523001", "Eja");

      System.out.println(mahasiswa1.toString());
      System.out.println(mahasiswa2.toString());
    }
}
```

```
run:
null null
null null
BUILD SUCCESSFUL (total time: 0 seconds)
```

- Contoh berikut adalah percobaan jika kata kunci **this** dihapus
- Program akan membaca nilai yang dimiliki oleh atribut kelas dibanding method
- Atribut kelas belum memiliki nilai

Overloading Method

Definisi

UNIVERSITAS ISLAM INDONESIA

- Mendeklarasikan lebih dari 1 (satu) method dengan kesamaan nama namun parameter yang berbeda.

Parameter berbeda adalah:

- Banyak parameter **berbeda**, tetapi tipe data **sama**
- Banyak parameter **sama**, tetapi tipe data **berbeda**
- Banyak parameter dan tipe datanya berbeda

UNIVERSITAS ISLAM INDONESIA

1

```
public class NamaClass {

    double hitung(double a, double b){
        return a*b;
    }

    double hitung(double a, double b, double c){
        return a*b;
    }
}
```

2

```
public class NamaClass {

    double hitung(double a, double b){
        return a*b;
    }

    float hitung(float a, float b){
        return a*b;
    }
}
```

3

```
public class NamaClass {

    double hitung(double a, double b){
        return a*b;
    }

    float hitung(float a, float b, float c){
        return a*b;
    }
}
```

(1)Banyak parameter berbeda, tipe data sama. (2) Banyak parameter sama, tipe data berbeda. (3) Banyak parameter dan tipe data berbeda.

* Dipelajari lebih lanjut di Polimorfisme

UNIVERSITAS
ISLAM
INDONESIA

Praktik

#1
Lanjutkan kode program agar menghasilkan output berikut

UNIVERSITAS ISLAM INDONESIA

```
Suara Kucing adalah Meong...
Suara Anjing adalah Guk guk...
PS D:\CODING\StudiKasus\StudiKasus>
```

```java
public class SuaraBinatang {
    public SuaraBinatang(String nama, String suara){
        System.out.println("Suara "+nama+" adalah "+suara);
    }

    Run | Debug
    public static void main(String[] args) {
        //TO DO something here
    }
}
```

#2
Modifikasi kode program agar menghasilkan output berikut

UNIVERSITAS ISLAM INDONESIA

```java
public class DemoThis {
    int a, b, c;

    DemoThis(int a, int b, int c){
        a = a;
        b = b;
        c = c;
    }

    void display(){
        System.out.println(a+" "+b+" "+c);
    }
}
```

File ke-1

```java
public class TesThis {
    Run | Debug
    public static void main(String args[]){
        DemoThis s1 = new DemoThis(111,222,333);
        DemoThis s2 = new DemoThis(444,555,666);

        //TO DO something here
    }
}
```

File ke-2

Output:

```
run:
111 222 333
444 555 666
BUILD SUCCESSFUL (total time: 0 seconds)
```

**Catatan: kedua file tersebut diletakkan pada folder yang sama*

UNIVERSITAS
ISLAM
INDONESIA

toString()

toString() Method

Contoh Implementasi

UNIVERSITAS ISLAM INDONESIA

- Jika ingin dibuat class yang modular (dapat digunakan tidak hanya di mode konsol, tetapi juga di mode GUI), sebaiknya **HINDARI** membuat pernyataan keluaran di dalam kelas.
- Contoh-contoh program yang di dalamnya ada pernyataan keluaran sekedar untuk memudahkan pemahaman mengenai pemanggilan method sehingga dalam praktik yang sebenarnya harus dimodifikasi dengan melibatkan method **toString()**
- Gunakan method **toString()** untuk gantinya & tampilkan nilai baliknya pada bagian yang memanggilnya.
- Method **toString()** digunakan untuk **mengembalikan** representasi string dari **objek**.

toString() Method
Contoh Implementasi

UNIVERSITAS ISLAM INDONESIA

```
public class DemoToString {
    String jurusan;
    int kode;
    String fakultas;
    String universitas;

    DemoToString(String jurusan, int kode, String fakultas,
                 String universitas)
    {
        this.jurusan = jurusan;
        this.kode = kode;
        this.fakultas = fakultas;
        this.universitas = universitas;
    }

    public String toString()
    {
        return jurusan + " " + kode + " " + fakultas + " "
             + universitas;
    }

    public static void main(String[] args)
    {
        DemoToString b = new DemoToString(
            "Informatika", 523, "FTI", "UII");

        System.out.println(b.toString());
    }
}
```

```
run:
Informatika 523 FTI UII
BUILD SUCCESSFUL (total time: 0 seconds)
```

UNIVERSITAS ISLAM INDONESIA

FUNDAMEN PENGEMBANGAN APLIKASI

Pertemuan 8

Terima Kasih