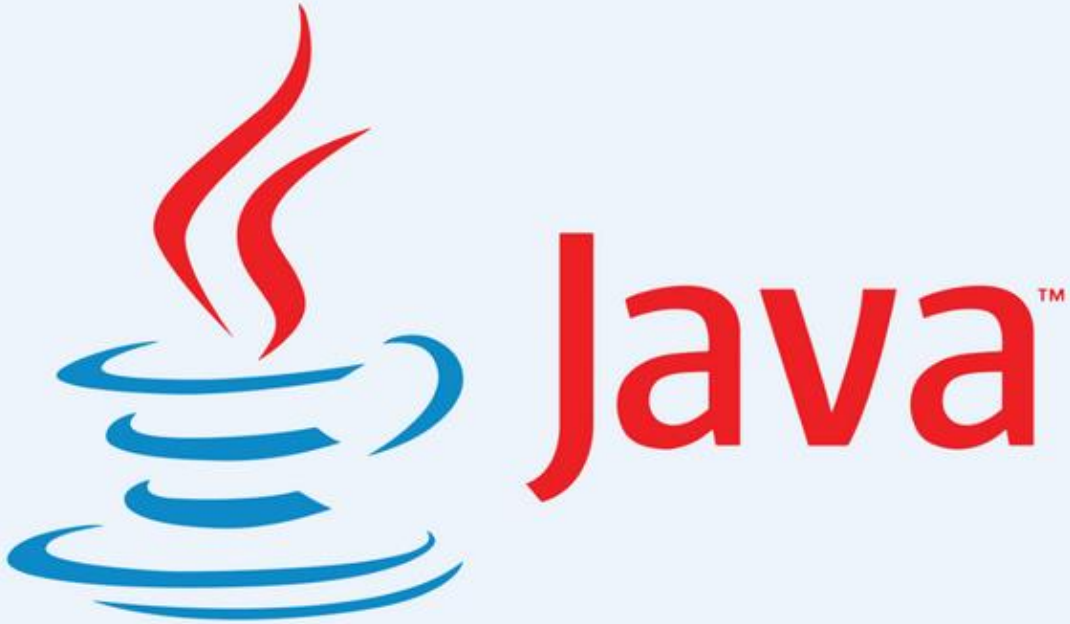
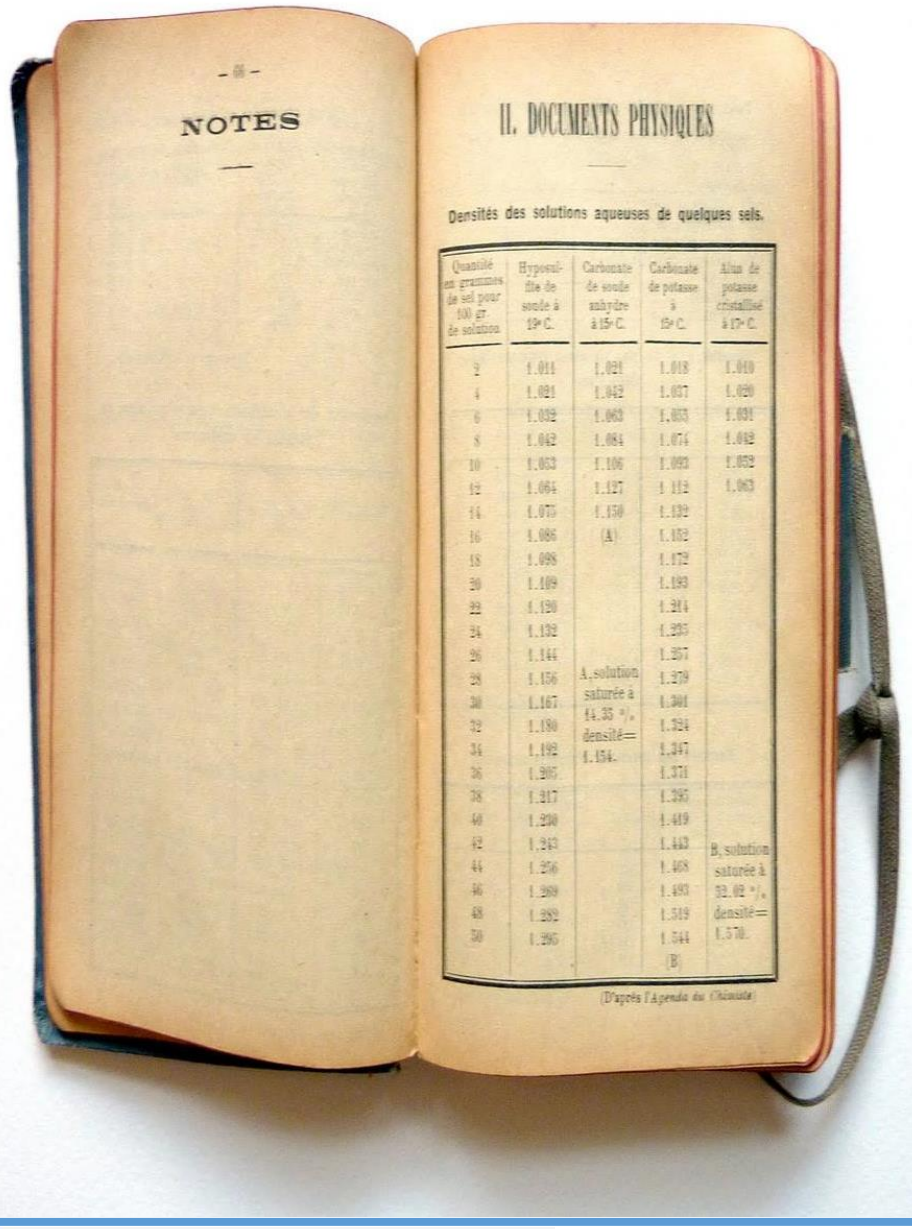
Pemrograman Berorientasi Objek



Pertemuan 3

Konsep Dasar Object & Class

Pemateri : Chrismikha Hardyanto S.Kom., M.Kom.



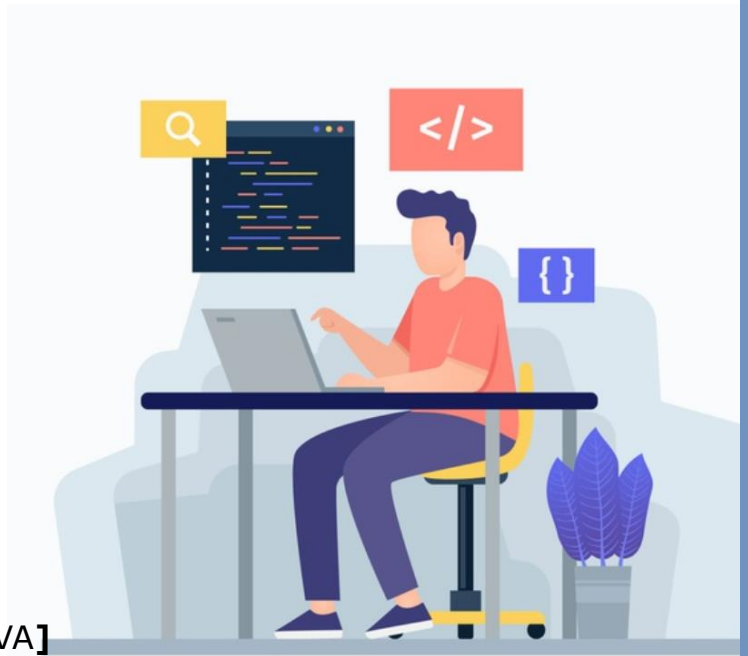
KONTEN PERKULIAHAN

- Apa itu Object dan Class didalam PBO
- Membuat Object dan Class dengan JAVA
- Member/Anggota sebuah class (Atribut & Method)
- Constructor
- Kata Kunci This

Konsep Dasar
Objek & Kelas

Kilas Balik Konsep PBO

Sebuah Paradigma/Pendekatan pemrograman yang menekankan pada **penciptaan objek** didalam menyelesaikan **masalah pemrograman**, dimana objek tersebut akan memiliki suatu **data** yang **merepresentasikan dirinya**



{OOP}

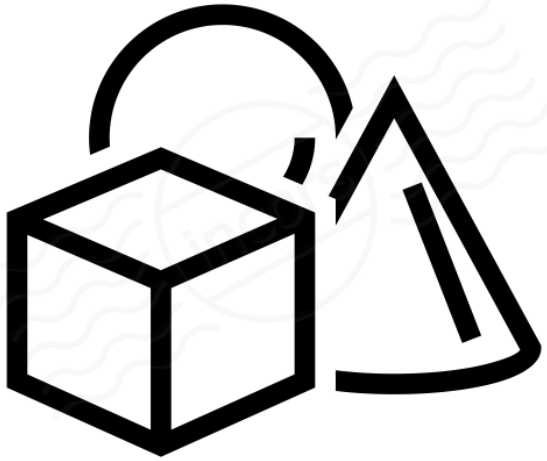
[sumber : Buku Pemrograman Berorientasi Objek JAVA**]**

Didalam PBO ada 2 istilah Penting yang selalu digunakan, yaitu :

Object dan Class

Apa itu Objek ?

- **Object** adalah suatu **data** yang berisi **atribut / field/ properties** dan **method / function/ behavior**
- **Atribut** merepresentasikan **karakteristik** (nilai) dari objek dan **method** merepresentasikan **sifat** (apa yang bisa dilakukan) dari objek
- semua data yang **bukan primitif di JAVA** adalah suatu objek (contohnya String)

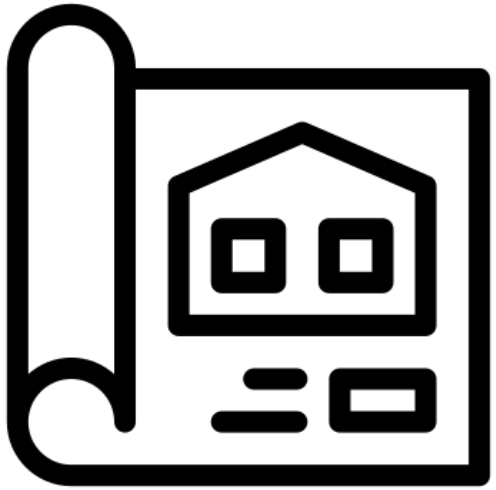


{OOP}

[sumber : Buku Pemrograman Berorientasi Objek JAVA**]**

Apa itu Class ?

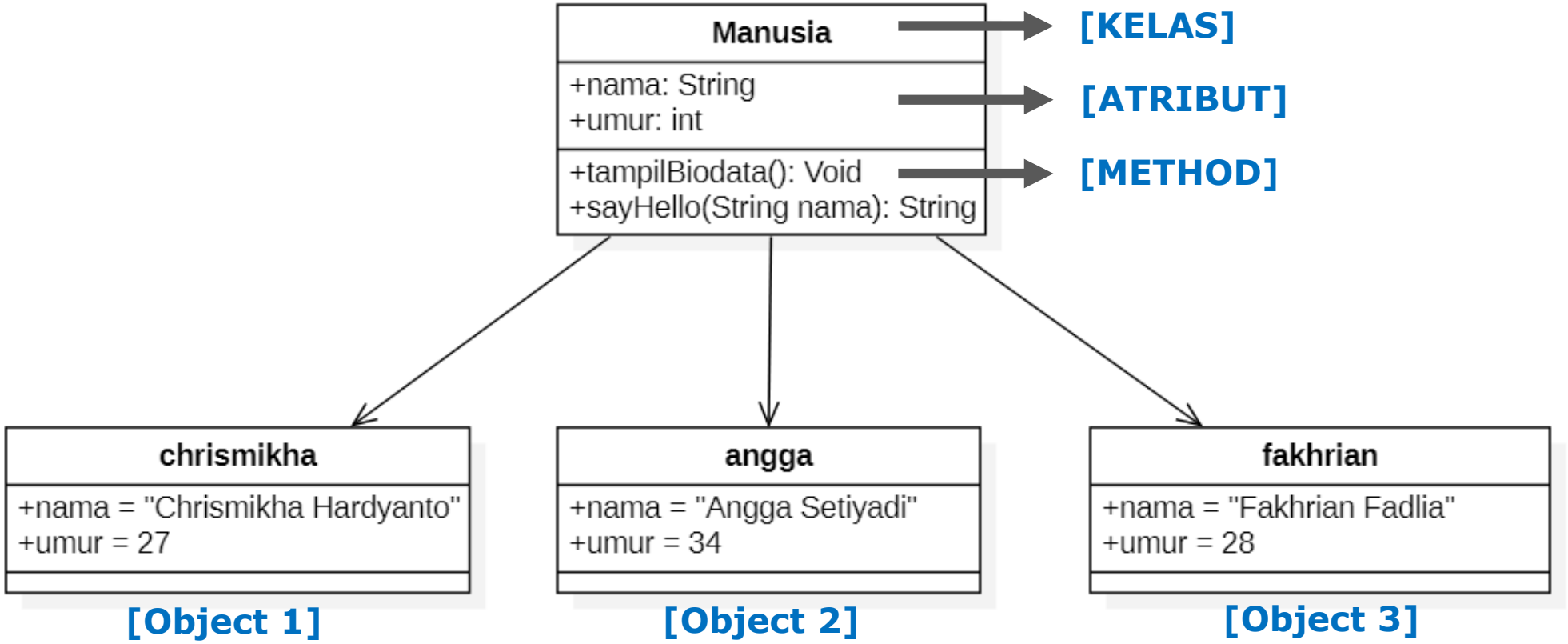
- **Class** adalah **blueprint, cetakan, prototype** untuk membuat Object
- Class berisikan **deklarasi** dari semua **atribut** dan **method** yang akan dimiliki oleh Object
- Setiap Object **selalu dibuat** dari Class. Dan sebuah Class bisa digunakan untuk membuat Object **tampa batas**



{OOP}

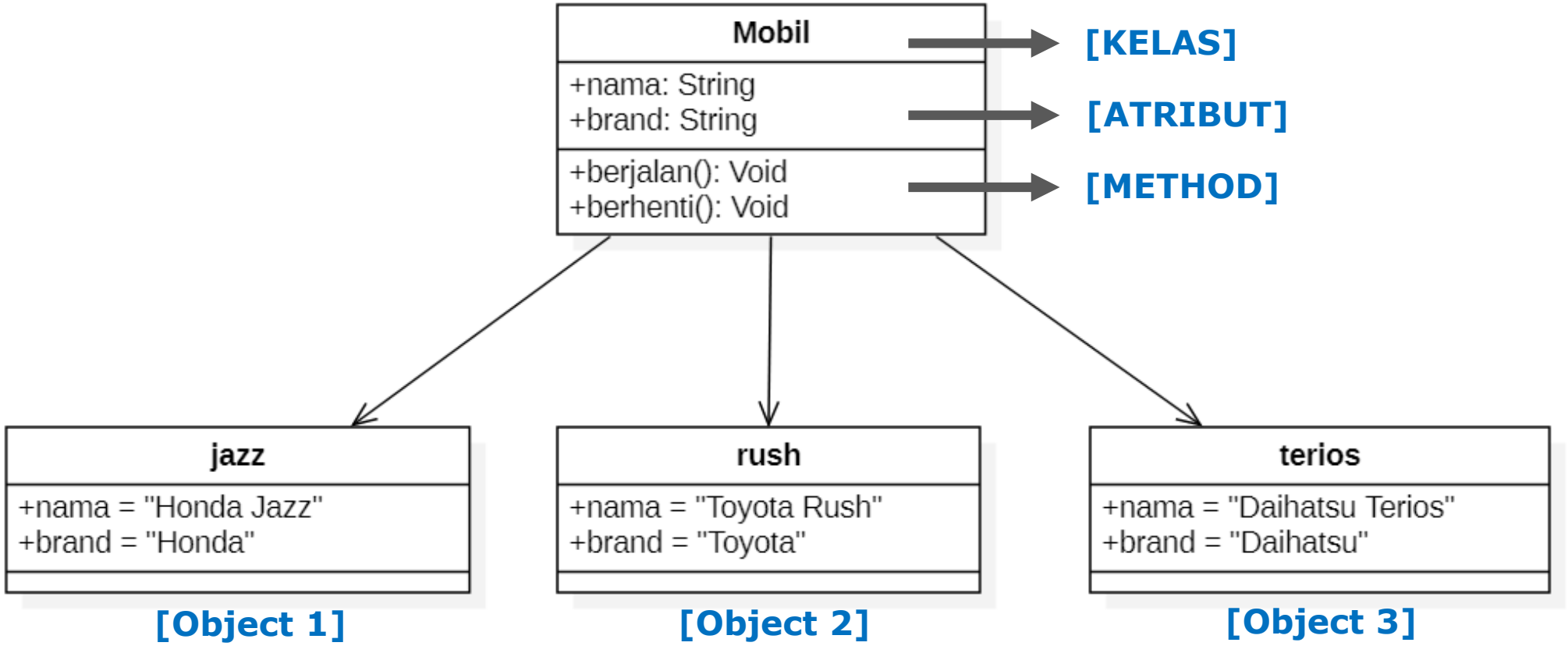
[sumber : Buku Pemrograman Berorientasi Objek JAVA**]**

Ilustrasi Konsep Object & Class



***Class** & **Object** Saling **Berhubungan**

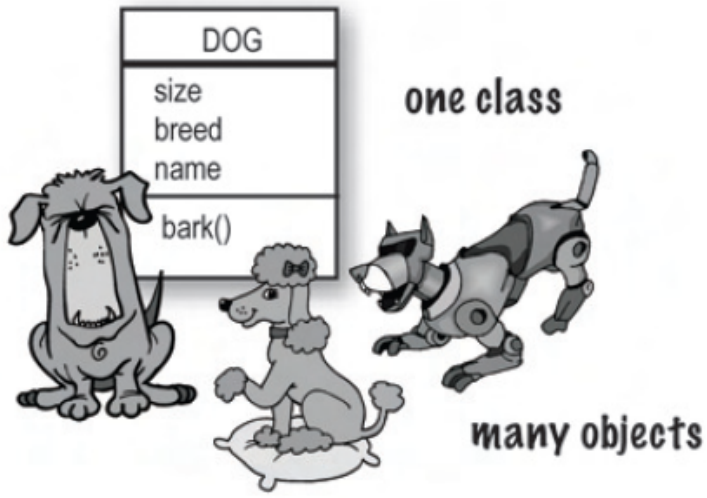
Ilustrasi Konsep Object & Class (2)



***Class** & **Object** Saling **Berhubungan**

Hubungan antara Object & Class

What's the difference between a class and an object?

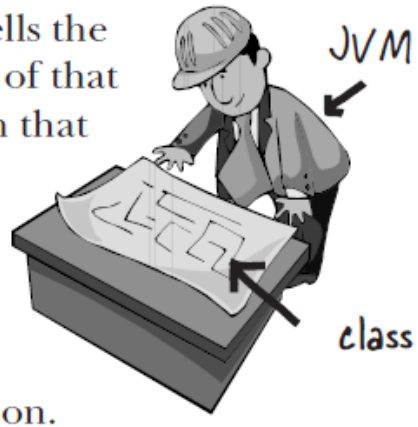


[**1 Kelas** = **N Buah Objek**]

A class is not an object.

(but it's used to construct them)

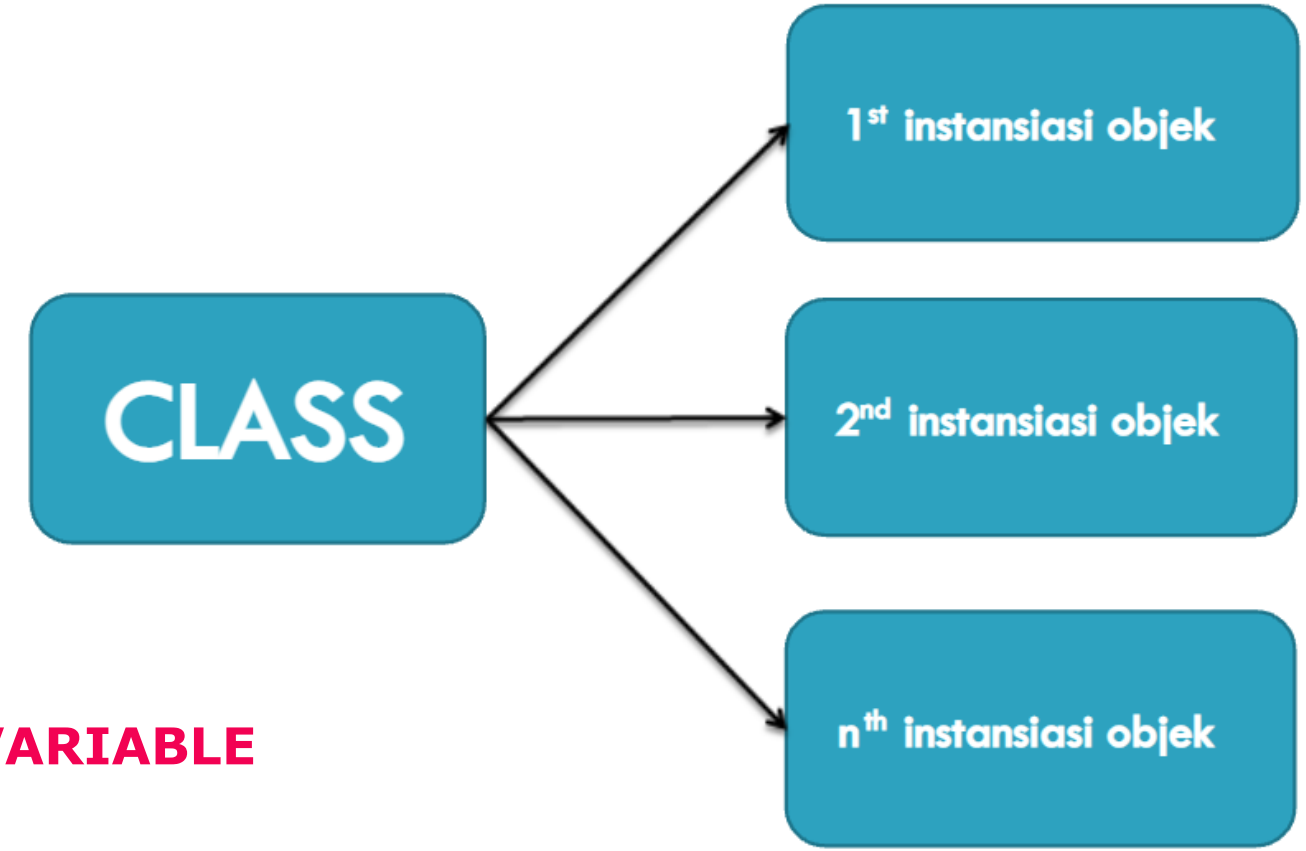
A class is a *blueprint* for an object. It tells the virtual machine *how* to make an object of that particular type. Each object made from that class can have its own values for the instance variables of that class. For example, you might use the Button class to make dozens of different buttons, and each button might have its own color, size, shape, label, and so on.



[sumber : head first JAVA 2nd ed**]**

Hubungan antara Object & Class

- Setiap object pasti memiliki **Class** (sebagai templatenya).
- Setiap object harus **diinstansiasi /dihidupkan** terlebih dahulu sebelum digunakan.
- **KELAS vs OBJEK** = **TIPE DATA vs VARIABLE**



Membuat Class pada JAVA

- Untuk membuat Class di JAVA, kita dapat menggunakan **kata kunci class**.
- Bentuk umum sintaksnya sebagai berikut

```
class NamaClass {
        //block kelas
}
```

- Penamaan Class umumnya mengunakan **format CamelCase** (Uppercase pada huruf pertama dan tidak boleh spasi) . Aturan nama dari class **sama dengan nama Variable**

[sumber : Buku Pemrograman Berorientasi Objek JAVA**]**

Implementasi Pada JAVA

```
class Manusia {
    /*Block Class - Digunakan untuk menulis isi member
      dari sebuah Class
    */


}
```

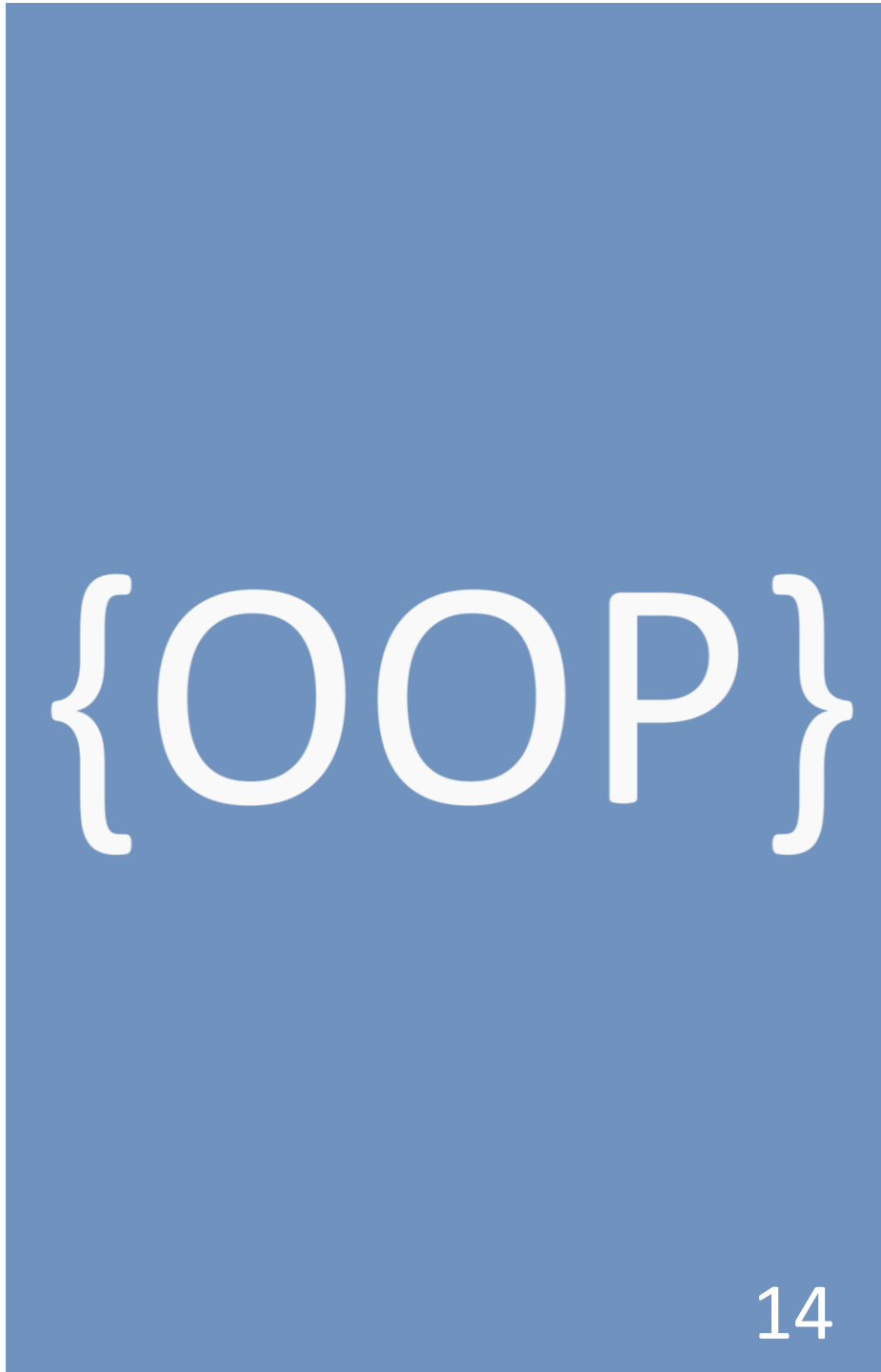
*Jika anda membuat **file class** pada netbeans **(file -> new -> java class)**, maka sintaks class akan dideklarasikan **otomatis**.

Membuat Object

- **Object** adalah hasil **instansiasi** dari sebuah ***Class***
- Untuk membuat object kita bisa menggunakan **kata kunci new**, dan diikuti dengan **nama Class** dan **tanda kurung ()**
- Ketika **menginstansiasi objek** umumnya references objek akan ditampung kedalam sebuah **variable** dengnan tipe data Object .
- Bentuk umum sintaksnya sebagai berikut

```
NamaClass namaObject = new NamaClass()
```

[sumber : Buku Pemrograman Berorientasi Objek JAVA**]**



Instansiasi Objek

The 3 steps of object declaration, creation and assignment

```
1        3   2
Dog myDog = new Dog();
```

1 Declare a reference variable

```
Dog myDog = new Dog();
```

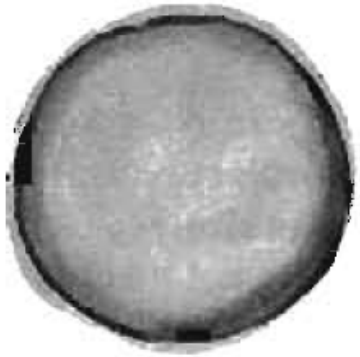
Tells the JVM to allocate space for a reference variable, and names that variable *myDog*. The reference variable is, forever, of type Dog. In other words, a remote control that has buttons to control a Dog, but not a Cat or a Button or a Socket.

Dog

2 Create an object

```
Dog myDog = new Dog();
```

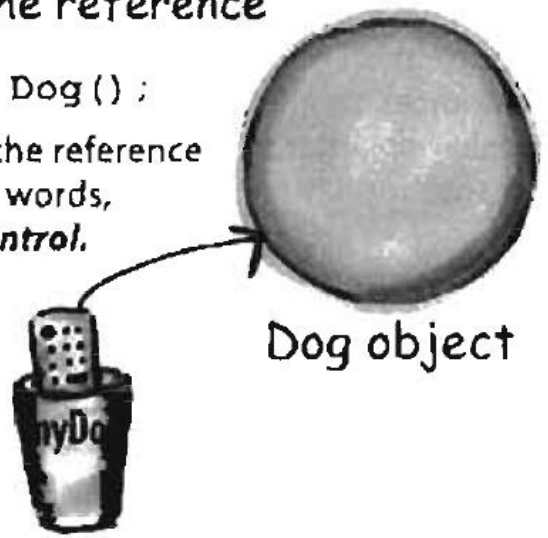
Tells the JVM to allocate space for a new Dog object on the heap (we'll learn a lot more about that process, especially in chapter 9.)


Dog object

3 Link the object and the reference

```
Dog myDog = new Dog();
```

Assigns the new Dog to the reference variable myDog. In other words, ***programs the remote control.***



Dog object

Dog

[sumber : head first JAVA 2nd ed**]**

Implementasi Pada JAVA

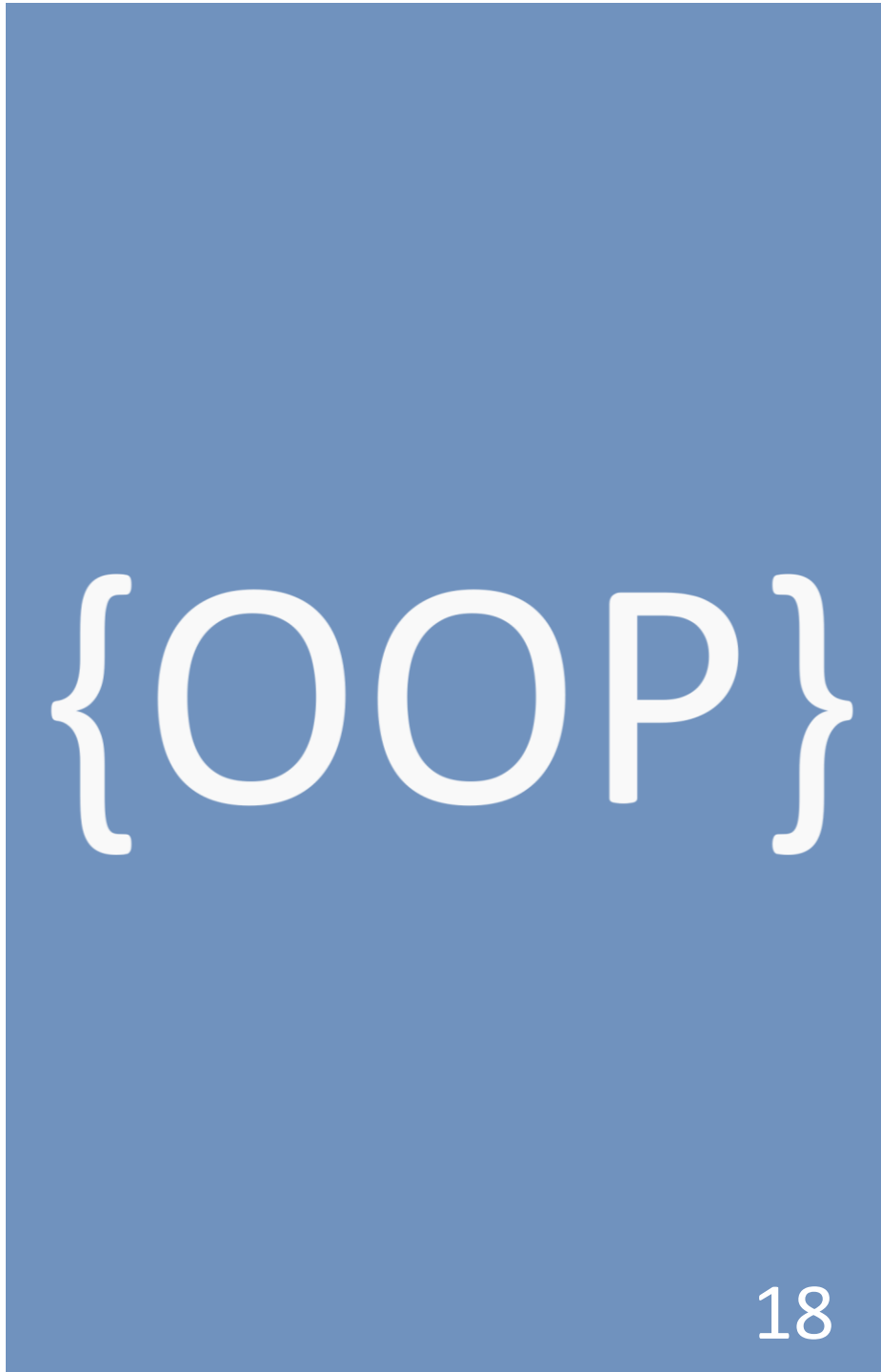
```
public static void main(String[] args) {

    //Intansiasi Object dari Class Manusia

    //Cara pertama
    Manusia chrismikha = new Manusia();

    //Cara kedua
    Manusia angga;
    angga = new Manusia();
}
```

*Lakukan instansiasi pbjek pada **main method** di main/tester class Anda

Sekarang kita sudah memahami konsep Class & Object, Dan bagaimana membuatnya di JAVA. Namun Object yang telah kita buat itu masih belum berguna

Atribut pada Objek

- **Atribut/Fields/Properties** adalah suatu **data** atau **nilai** yang bisa kita sisipkan didalam Object.
- Sebelum kita bisa memasukan nilai kedalam atribut, kita harus **mendeklarasikan data** apa saja yang dimiliki oleh Object tersebut didalam deklarasi **class-nya**
- Membuat atribut **sama** seperti **membuat variable**, namun ditempatkan didalam **block Class**. (Dikenal juga dengan istilah instance variable)



[sumber : Buku Pemrograman Berorientasi Objek JAVA**]**

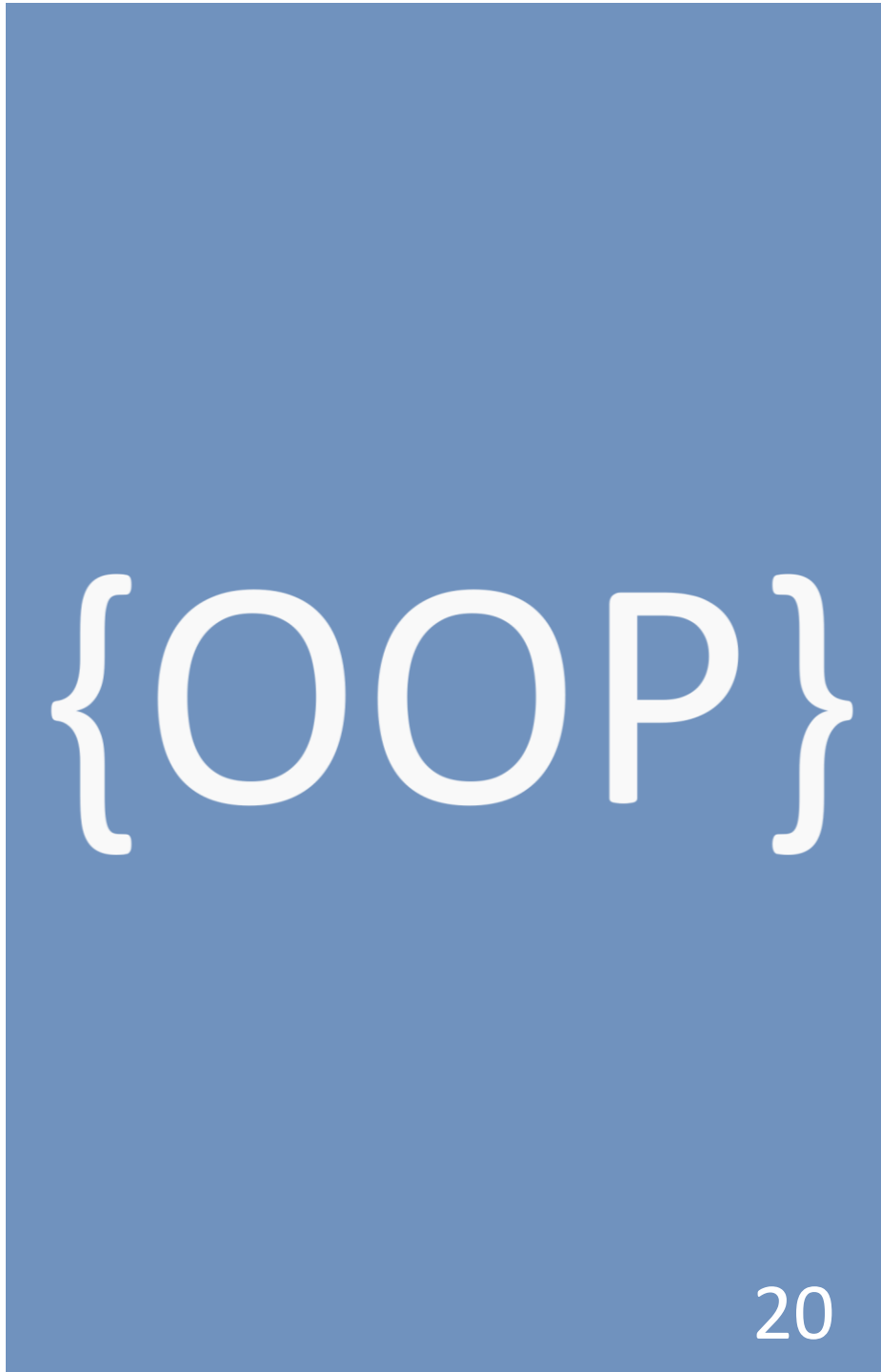
Implementasi Atribut Pada JAVA

```
class Manusia {
    /*Block Class - Digunakan untuk menulis isi member  dari sebuah Class */
    /*Deklarasi Atribut dari Object didalam Class*/

    String nama;
    String alamat;
    int umur;
    final String NEGARA = "Indonesia";

}
```

*Lakukan instansiasi objek pada **main method** di main/tester class Anda

Manipulasi Atribut

- Atribut yang ada pada Object dapat kita **manipulasi (mengisi/mengubah)** datanya. Tergantung dia final (konstanta) atau bukan.
- Jika **final**, berarti kita **tidak bisa** mengubah data atributnya. Namun jika tidak, kita bisa mengubah isi atributnya.
- **Cara memanipulasi** data/nilai didalam atribut, sama seperti cara memanipulasi data **pada variable**.
- Untuk mengakses suatu atribut, gunakan **kata kunci .** (**titik**) setelah **nama object** dan diikuti n**ama atributnya**.



Manipulasi Atribut dari Object

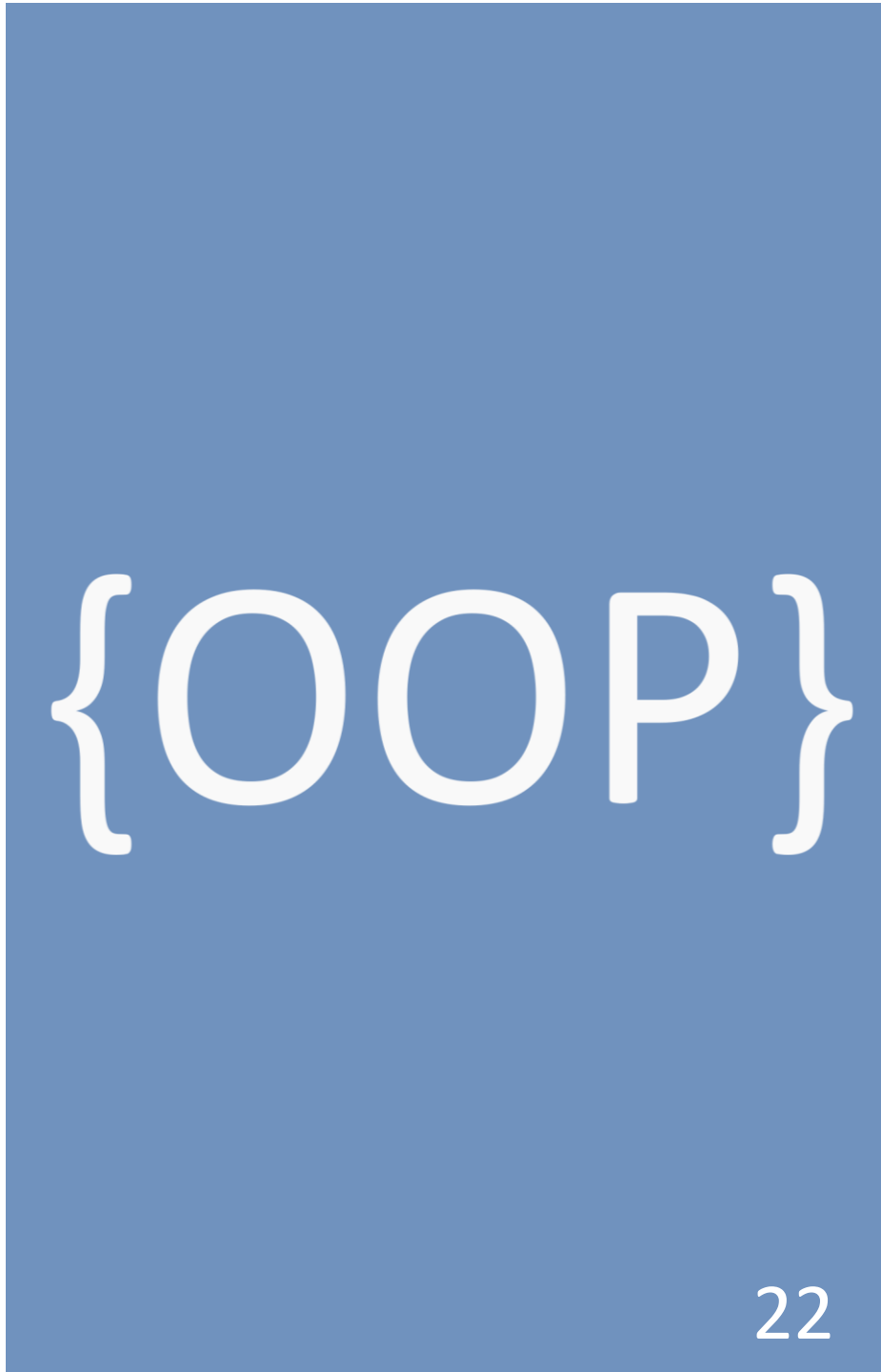
```
public static void main(String[] args) {

    //Intansiasi Object dari Class Manusia

    Manusia chrismikha = new Manusia();

     //Mengisi nilai kedalam atribut pada object chrismikha
    chrismikha.nama="Chrismikha Hardyanto";
    chrismikha.alamat = "Margahayu Raya";
    chrismikha.umur = 27;
    //chrismikha.NEGARA = Tidak bisa dimanipulasi karena final

    //Menampilkan isi dari setiap atribut pada object chrismikha
    System.out.println("Nama      : " +chrismikha.nama);
    System.out.println("Ulamat    : " +chrismikha.alamat);
    System.out.println("Umur      : " +chrismikha.umur);

}
```

```
run:
Nama      : Chrismikha Hardyanto
Ulamat    : Margahayu Raya
Umur      : 27
```

*hasil RUN program

Method pada Objek

- Selain menambahkan atribut, Kita juga bisa **menambahkan method** kedalam object. Cara mendeklarasikan method adalah **didalam block class-nya**.
- Method pada JAVA juga terbagi menjadi 2 jenis. **Method tampa nilai balik** & **Method dengan nilai balik**.
- Untuk mengakses method, kita bisa menggunakan **tanda .**(**titik**) dan diikuti dengan **nama method** nya. (Sama seperti **cara mengakses atribut**).



Deklarasi Method pada JAVA (2)

- Bentuk umum sintaks untuk medeklarasikan method pada Java adalah sebagai berikut :

Tampa Nilai Balik :

```
<void> namaMethod(<tipe data> parameter1, ..., ke-n) {

        //block Method, berisi seluruh kode program milik method;
}
```

Dengan Nilai Balik :

```
<tipe data> namaMethod(<tipe data> parameter1, ..., ke-n) {

        //block Method, berisi seluruh kode program milik method;
        return <nilai yang dibalikan>
}
```

*[Di JAVA hanya bisa **membalikan** sebuah nilai saja]

Implementasi Method Pada JAVA

```
/*Deklarasi Method dari Object didalam Class*/
void tampilBiodata(){
    System.out.println("Nama      : " +nama);
    System.out.println("Ulamat    : " +alamat);
    System.out.println("Umur      : " +umur);
}

String sayHello(String paramNama){
    String tampilSalam = "Halo " +paramNama+" Nama Saya " +nama;
    System.out.println(tampilSalam);
    return tampilSalam;
}
```

*Lakukan instansiasi objek pada **main method** di main/tester class Anda untuk dapat mengakses method didalam class

Implementasi Method Pada JAVA

```
//memanggil method
chrismikha.tampilBiodata();
System.out.println("");
chrismikha.sayHello("Eko");
```

Mengakses method tampiBiodata() tampa parameter

Mengakses method sayHello() dengan 1 parameter bertipe String

```
run:
Nama      : Chrismikha Hardyanto
Ulamat    : Margahayu Raya
Umur      : 27

Halo Eko Nama Saya Chrismikha Hardyanto
BUILD SUCCESSFUL (total time: 0 seconds)
```

Rangkuman membuat Object & Class

1 Write your class

```
class Dog {

  int size;
  String breed;
  String name;

  void bark() {
    System.out.println("Ruff! Ruff!");
  }
}
```

instance variables

a method

2 Write a tester (TestDrive) class

just a main method

(we're gonna put code in it in the next step)

```
class DogTestDrive {
  public static void main (String[] args) {
    // Dog test code goes  here
  }
}
```

3 In your tester, make an object and access the object's variables and methods

```
class DogTestDrive {
  public static void main (String[] args) {
    Dog d = new Dog();
    d.size = 40;
    d.bark();
  }
}
```

make a Dog object

use the dot operator (.) to set the size of the Dog

and to call its bark() method

dot operator

[**sumber :** head first JAVA 2nd ed]

Setelah Anda memahami bagaimana membuat Class & Object pada JAVA. Selanjutnya kita lihat beberapa konsep PBO lain yang mendukung konsep Class & Object

Construktor pada JAVA

- Saat kita membuat/instansiasi sebuah Object dari Class, maka kita seperti memanggil sebuah method karena menggunakan kurung(). Itu disebut sebagai **construktor**
- Didalam Class JAVA (konsep PBO secara umum), Construktor adalah **method khusus yang akan dipanggil saat pertama kali Object dibuat**.
- Fungsi Construktor mirip seperti method pada umumnya, Kita juga bisa memberikan parameter.
- Satu syarat penting dalam pembuatan construktor adalah **namanya harus sama** dengan **nama Class**, dan tidak membutuhkan kata kunci **void** atau **return value**

Implementasi Construktor

[Sintaks dasar pembuatan Contrusktor]

```
//Deklarasi Construktor pada Class Manusia
Manusia(){
    //Block Construktor

}
```

[contoh constructor untuk inisialisasi **nilai awal** pada object]

```
Manusia(String paramNama, String paramAlamat, int paramUmur){
    //Block Construktor
    nama = paramNama;
    alamat = paramAlamat;
    umur = paramUmur;
}
```

[contoh pemanggilan constructor pada main Class]

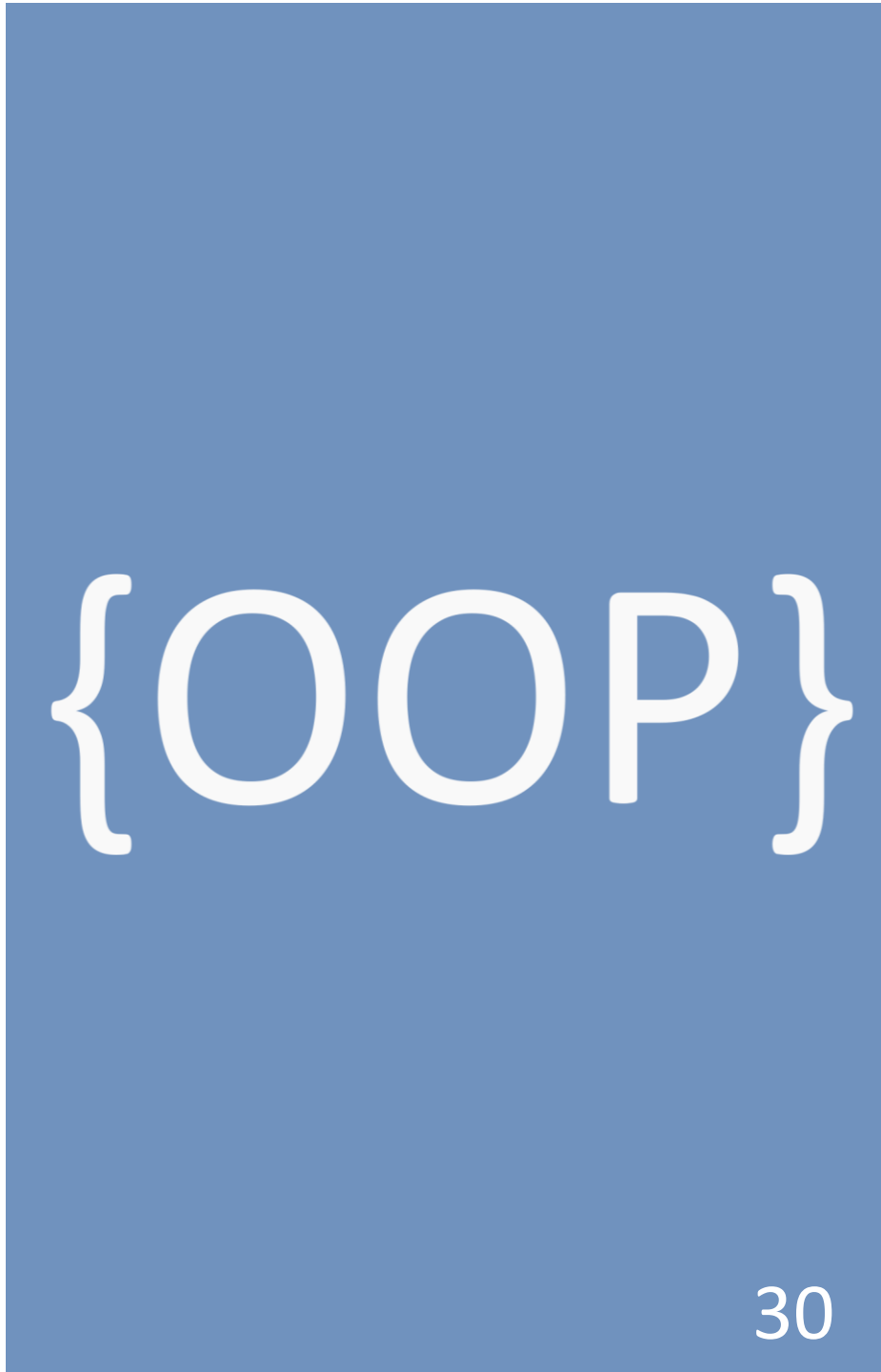
```
Manusia chrismikha;
chrismikha = new Manusia("Chrismikha Hardyanto","Margahayu Raya",27);

 //Mengisi nilai kedalam atribut pada object chrismikha
chrismikha.nama="Chrismikha Hardyanto";
chrismikha.alamat = "Margahayu Raya";
chrismikha.umur = 27;
//chrismikha.NEGARA = Tidak bisa dimanipulasi karena final
```

Coba Anda hapus baris perintah kode program untuk mengisi nilai pada objek dari kelas Manusia lalu run program Anda, apa yg terjadi….

*Kode program didalam constructor akan **otomatis dieksekusi** ketika objek dibuat. Tidak seperti method pada umumnya yang perlu diakses dulu dari suatu object

Variable Shadowing

- **Variable shadowing** adalah kejadian ketika kita membuat nama variable dengan **nama yang sama** di scope yang **menutupi variable dengan nama yang sama pada scope diatasnya** .
- Masalah ini biasa terjadi seperti saat kita membuat **nama parameter** didalam method dan ternyata **namanya sama dengan nama atribut** didalam **kelas** tempat method tersebut dideklarasikan
- Ketika terjadi variable shadowing, maka secara otomatis variable pada scope diatasnya **tidak bisa diakses.**



Contoh kasus Pada JAVA

```java
class Manusia {
    /*Block Class - Digunakan untuk menulis isi member  dari sebuah Class */
    /*Deklarasi Atribut dari Object didalam Class*/

    String nama;
    String alamat;
    int umur;
    final String NEGARA = "Indonesia";

    //Deklarasi Construktor pada Class Manusia
    Manusia(String nama, String alamat, int umur){
        //Block Construktor
        nama = nama;
        alamat = alamat;
        umur = umur;
    }

    /*Deklarasi Method dari Object didalam Class*/
    void tampilBiodata(){
        System.out.println("Nama      : " +nama);
        System.out.println("Ulamat    : " +alamat);
        System.out.println("Umur      : " +umur);
    }

    String sayHello(String nama){
        String tampilSalam = "Halo " +nama+" Nama Saya " +nama;
        System.out.println(tampilSalam);
        return tampilSalam;
    }
}
```

*Coba ubah penulisan **nama parameter** di program anda sebelumnya pada kelas manusia

```
run:
Nama      : Chrismikha Hardyanto
Ulamat    : Margahayu Raya
Umur      : 27

Halo Eko Nama Saya Eko
BUILD SUCCESSFUL (total time: 0 seconds)
```

*Jika Anda run programnnya, maka akan **ada yang aneh** dari output program sebelumnya

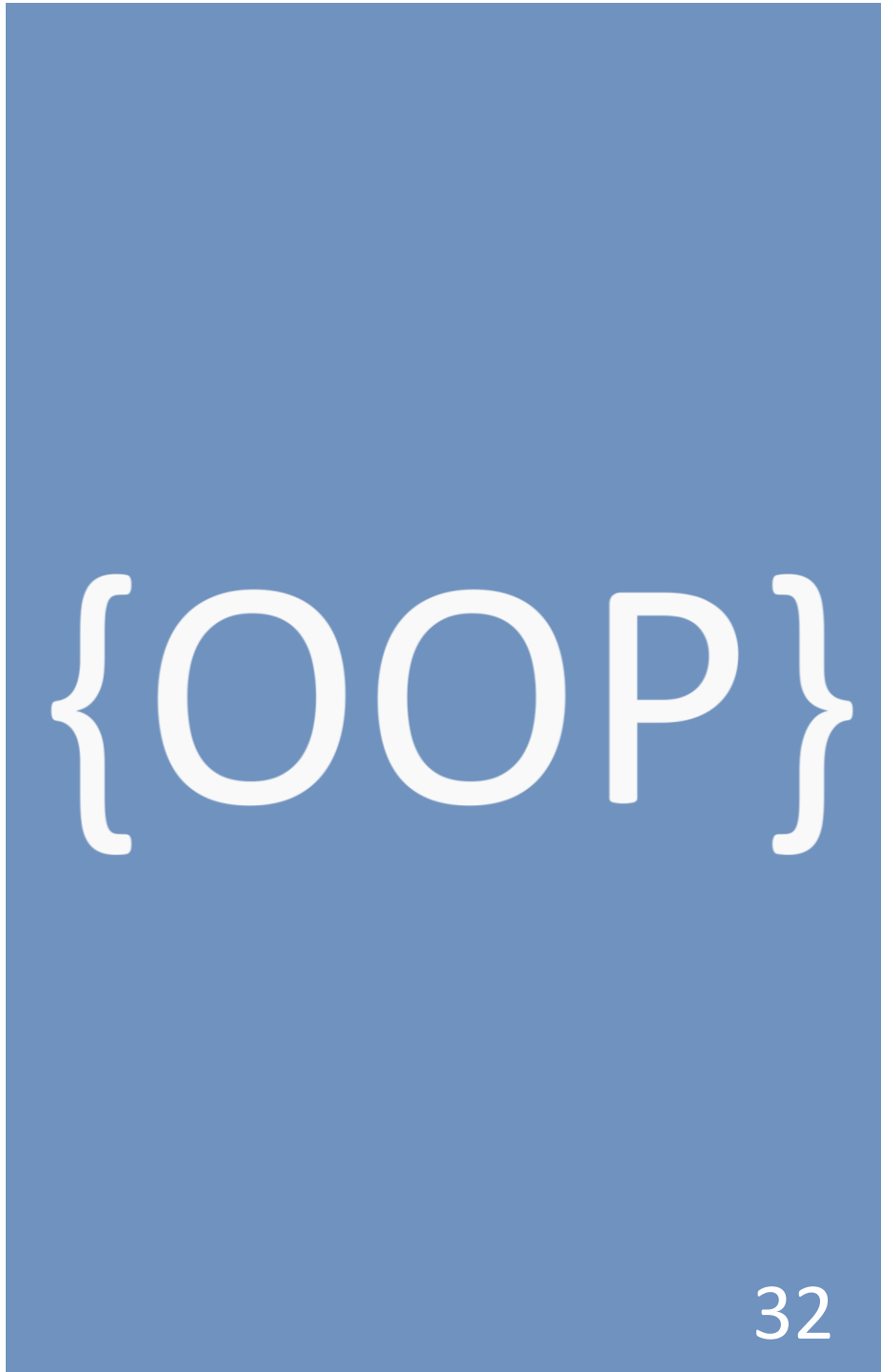
SOLUSINYA ? Gunakan kata kunci this

Kata kunci this

- Saat kita membuat kode didalam block construktor atau method didalam class, kita bisa menggunkan k**ata kunci this** untuk **mengakses object saat ini**

- Misalkan kita butuh mengakses sebuah atribut yang namanya sama dengan parameter method, agar **tidak tertukar (variable shadowing)** kita bisa menggunakan kata kunci this pada untuk mengakses atribut tersebut.

- kata kunci this tidak hanya digunakan untuk mengakses atribut milik objek saat ini, namun juga bisa digunakan mengakses method



Contoh kasus Pada JAVA

```
class Manusia {
    /*Block Class - Digunakan untuk menulis isi member  dari sebuah Class */
    /*Deklarasi Atribut dari Object didalam Class*/

    String nama;
    String alamat;
    int umur;
    final String NEGARA = "Indonesia";

    //Deklarasi Construktor pada Class Manusia
    Manusia(String nama, String alamat, int umur){
        //Block Construktor
        this.nama = nama;
        this.alamat = alamat;
        this.umur = umur;
    }

    /*Deklarasi Method dari Object didalam Class*/
    void tampilBiodata(){
        System.out.println("Nama      : " +this.nama);
        System.out.println("Ulamat    : " +this.alamat);
        System.out.println("Umur      : " +this.umur);
    }

    String sayHello(String nama){
        String tampilSalam = "Halo " +nama+" Nama Saya " +this.nama;
        System.out.println(tampilSalam);
        return tampilSalam;
    }
}
```

*tambahkan kata kunci **this** untuk **setiap pemanggilan atribut** didalam method pada kelas Manusia

```
run:
Nama      : Chrismikha Hardyanto
Ulamat    : Margahayu Raya
Umur      : 27

Halo Eko Nama Saya Chrismikha Hardyanto
BUILD SUCCESSFUL (total time: 0 seconds)
```

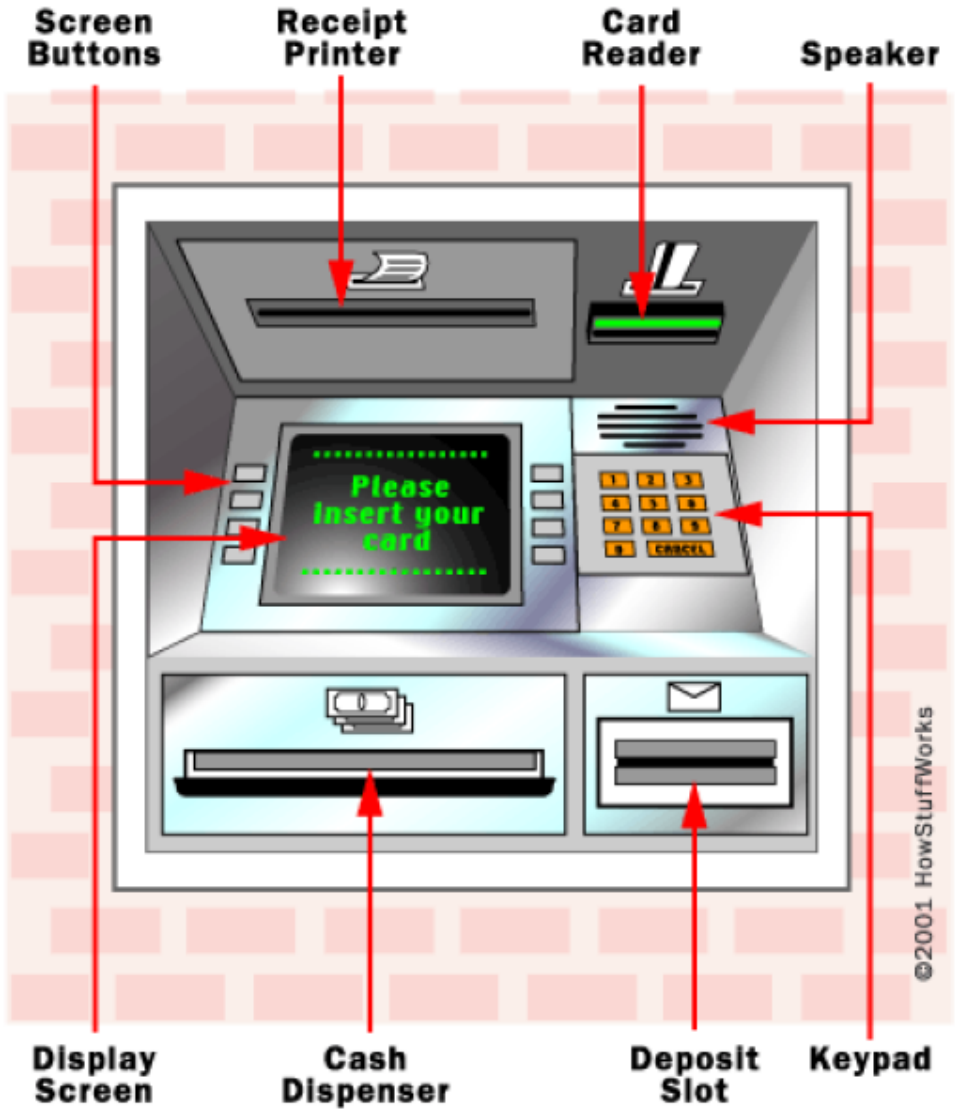
Ketika program di run, akan dapat berjalan sesuai harapan

Setelah memahami pembuatan class & Objek pada JAVA, mari kita berlatih

*Silakan buka modul praktikum pertemuan 4

Contoh Menemukan Objek

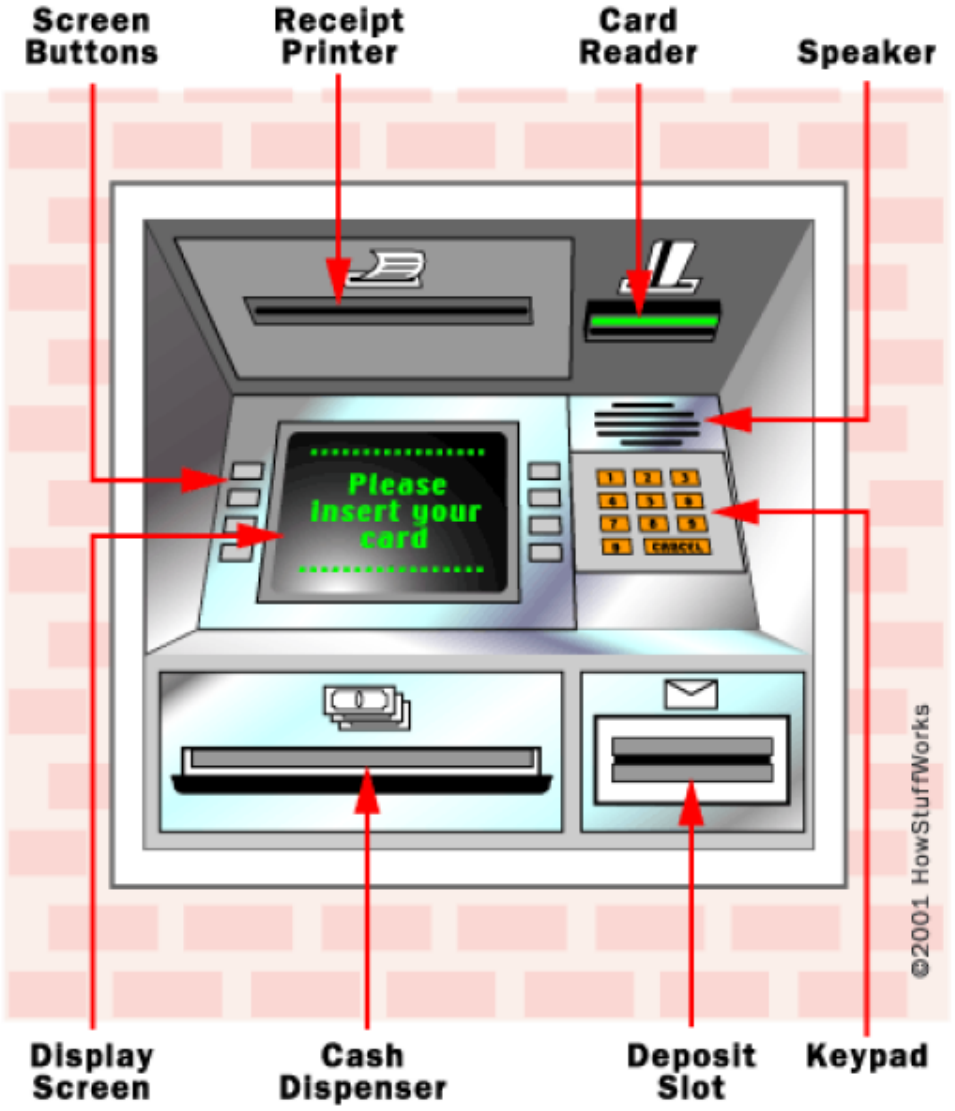
- Perhatikan mesin ATM
- Sebuah mesin ATM terdiri dari elemen-elemen berikut :
  - Display Screen (Layar)
  - Screen Button (Tombol)
  - Receipt Printer
  - Card Reader
  - Speaker
  - Cash Dispenser
  - Deposit Slot
  - Keypad
- Di dalam konsep OOP, setiap setiap elemen tersebut berinteraksi dengan mengirimkan pesan (message) tertentu.



[**sumber** : Modul PBO – Andri Heryandi,M.T]

Contoh Menemukan Objek

- Interaksi ketika penekanan tombol Ambil Uang Rp. 200.000
  - Speaker mengeluarkan bunyi beep.
  - CashDispenser mengeluarkan uang 200.000
  - Jika uang diambil, maka Receipt Printer mencetak faktur,
  - Jika uang tidak diambil, Uang dimasukkan kembali ke Cash Dispenser,
  - Layar kembali ke menu Utama.



[**sumber** : Modul PBO – Andri Heryandi,M.T]

Terima Kasih