ENKAPSULASI

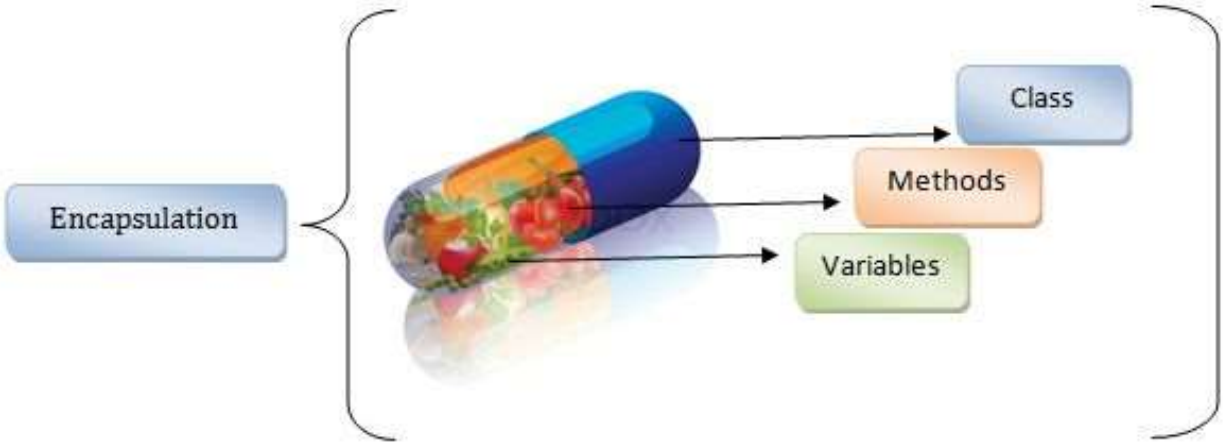
Minggu ke-3 PBO

Overview

- Enkapsulasi
- Konstruktor
- Akses Modifier
- Atribut/method pada class
- Intansiasi atribut/method
- Setter dan getter
- Memahami notasi pada UML Class Diagram

Enkapsulasi

- Information-hiding



- Misalnya vacum cleaner, ketika kita mencolokkan kabel vacum cleaner dan menyalakan sakelarnya maka mesin tersebut siap digunakan untuk menghisap debu. Dalam proses tersebut kita tidak mengetahui proses rumit yang terjadi ketika mengubah listrik menjadi tenaga dari vacum cleaner.

Enkapsulasi

- **Enkapsulasi** adalah salah satu yang paling terpenting di dalam pemrograman berorientasi objek(OOP).
- Tahap enkapsulasi menentukan pemberian hak akses pada setiap property atau method.

Enkapsulasi

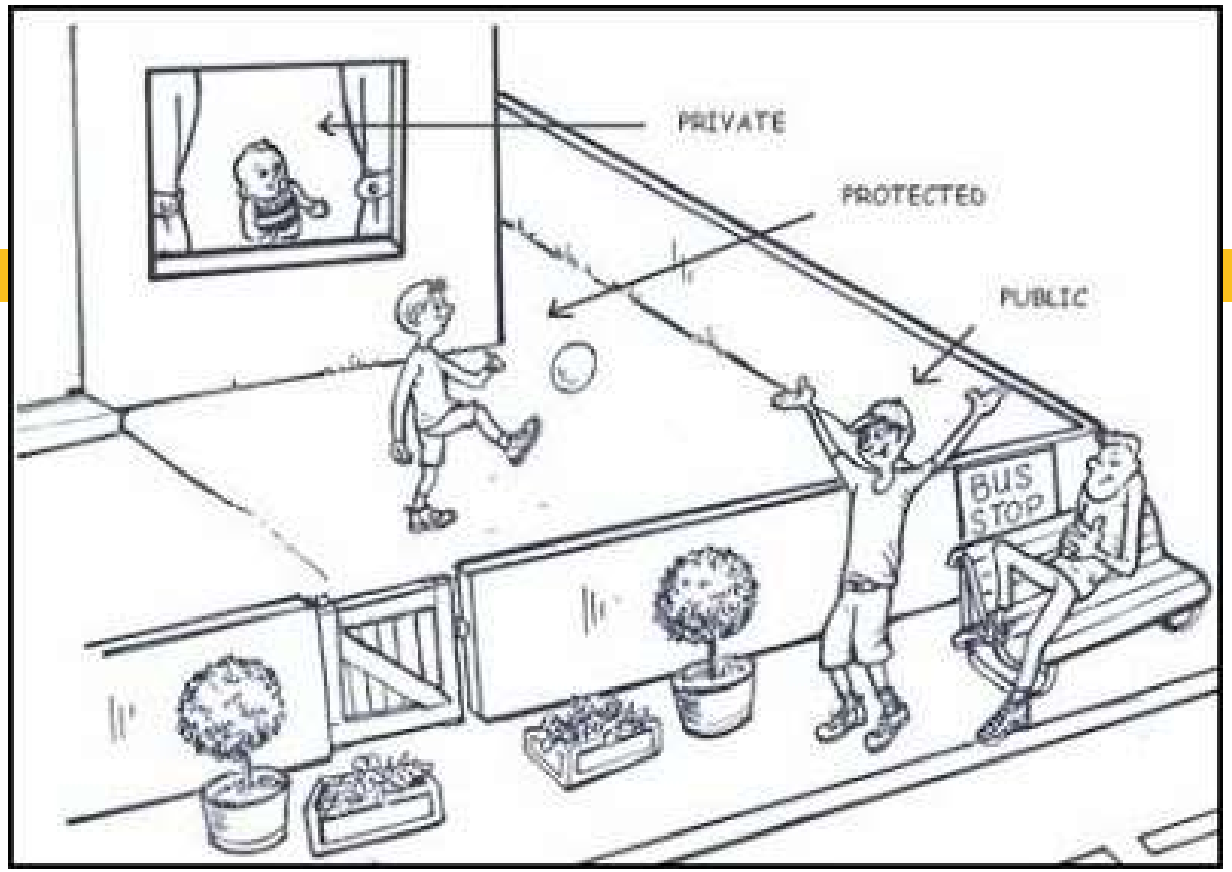
```
package motorencapsulation;
public class MotorDemo {
    public static void main(String[] args) {
        Motor motor = new Motor();
        motor.printStatus();
        motor.tambahKecepatan();

        motor.nyalakanMesin();
        motor.printStatus();
  }
}
```

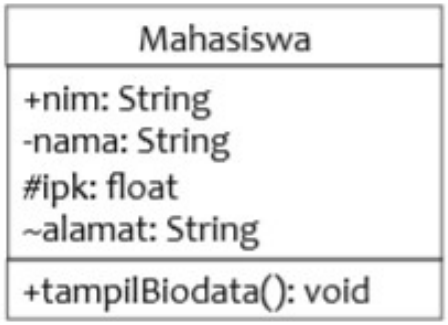
- Disini user tidak mengetahui proses yang ada di dalam method printStatus(), tambahKecepatan()

Akses Modifier

- Terdapat 4 akses modifier yaitu:
- 1. Private – hanya dapat diakses di dalam kelas yang sama
  2. Default – hanya dapat diakses di dalam package yang sama
  3. Protected – hanya dapat diakses oleh kelas yang sama, package yang sama, dan kelas turunannya (*subclass*)
- 4. Public – dapat diakases dari mana saja



Modifier	class yang sama	package yang sama	subclass package lain	class manapun
private	√			
default	√	√		
protected	√	√	√	
public	√	√	√	√




```
class Mahasiswa{
    public String nim;
    private String nama;
    protected float ipk;
    String alamat;

    public void tampilBiodata(){
        ...
    }
}
```

+ public
- private
protected
~ default

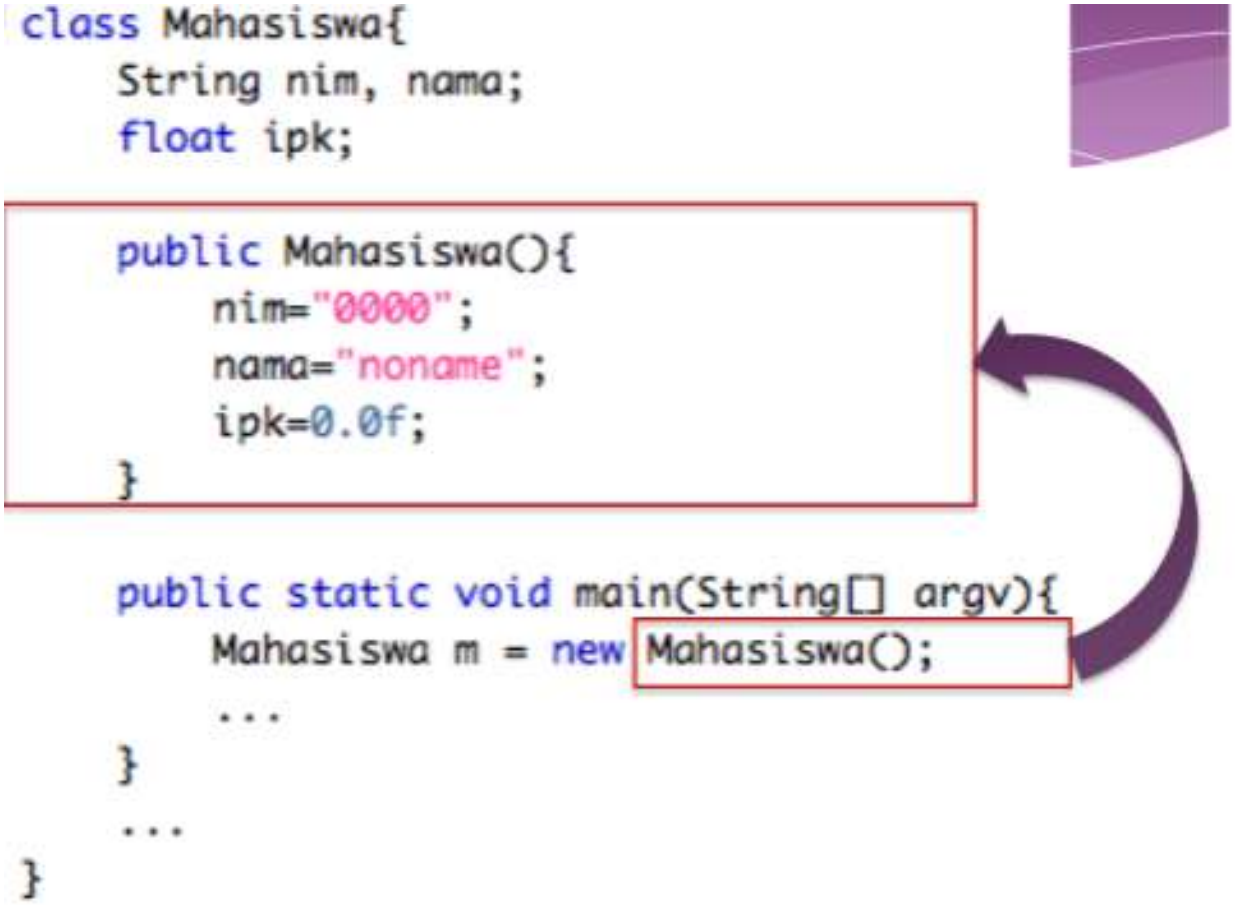
Konstruktor

- Dieksekusi pertama kali ketika instan dari objek dibuat
- Cara untuk membuat konstruktor adalah sebagai berikut:

A. **Nama konstruktor harus sama dengan nama class**
B. Selalu memiliki modifier access public
C. Konstruktor hanya jalan saat proses instansiasi
D. Konstruktor tidak memiliki tipe data method
E. Tidak bisa memiliki return

Konstruktor

Contoh



```
class Mahasiswa{
    String nim, nama;
    float ipk;

    public Mahasiswa(){
        nim="0000";
        nama="noname";
        ipk=0.0f;
    }

    public static void main(String[] argv){
        Mahasiswa m = new Mahasiswa();
        ...
    }
    ...
}
```

Instansiasi Atribut / Method

- Atribut dan method instansiasi merupakan **atribut dan method yang dimiliki oleh objek hasil instansiasi.**
- Untuk akses atribut dan method sebelumnya harus membuat objek terlebih dahulu.
- Cara akses atribut maupun method instansiasi, harus terlebih dahulu membuat objek

```
.....
//lakukan dulu instansiasi objek
nama_objek.nama_atribut
nama_objek.nama_method();
```

```
Mahasiswa m = new Mahasiswa();
m.nim = "0210630064";
m.nama = "Imam F";
m.ubahNama("Imam Fahrur Rozi")
```

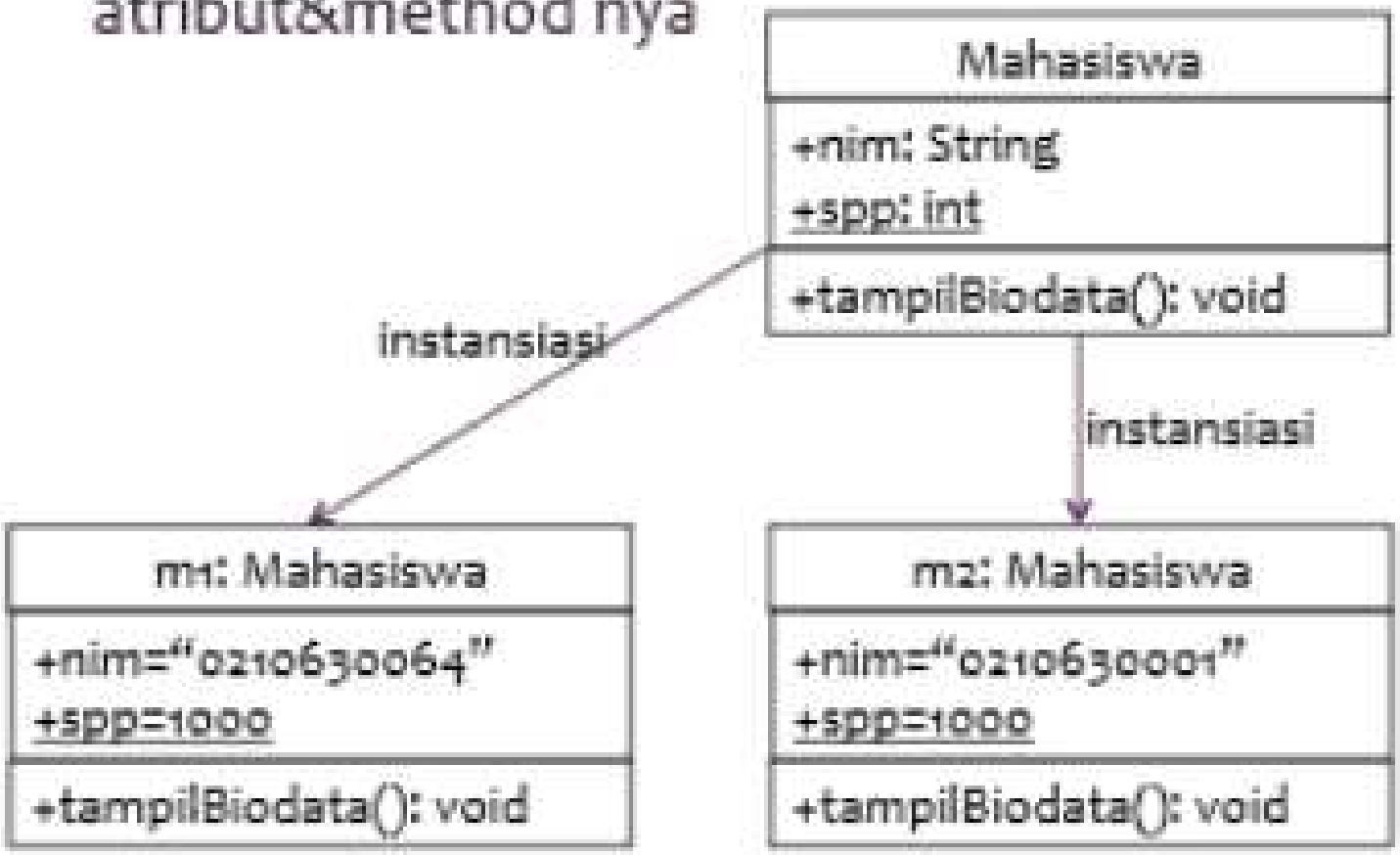
Atribut / Method pada Class

- Atribut dan method dimiliki oleh Class
- Bernilai sama untuk semua objek dari class yang sama
- Atribut dan method class dideklarasikan menggunakan kata kunci **static**
- Cara akses atribut maupun method class:

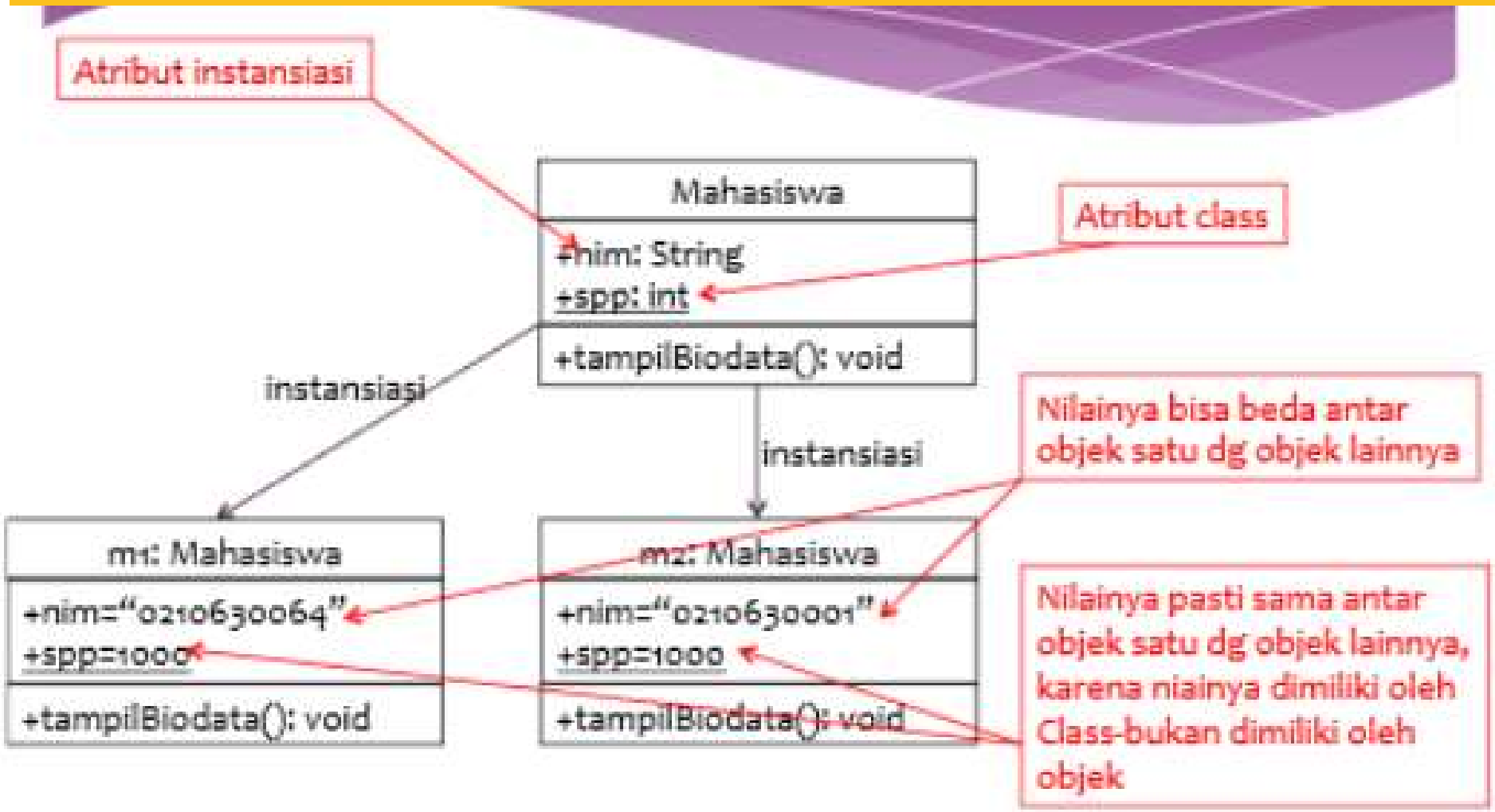
```
//tidak perlu instansiasi, langsung menggunakan
//nama class
NamaKlas.namaAtribut;
NamaKlas.namaMethod();
```

Atribut dan Method Class

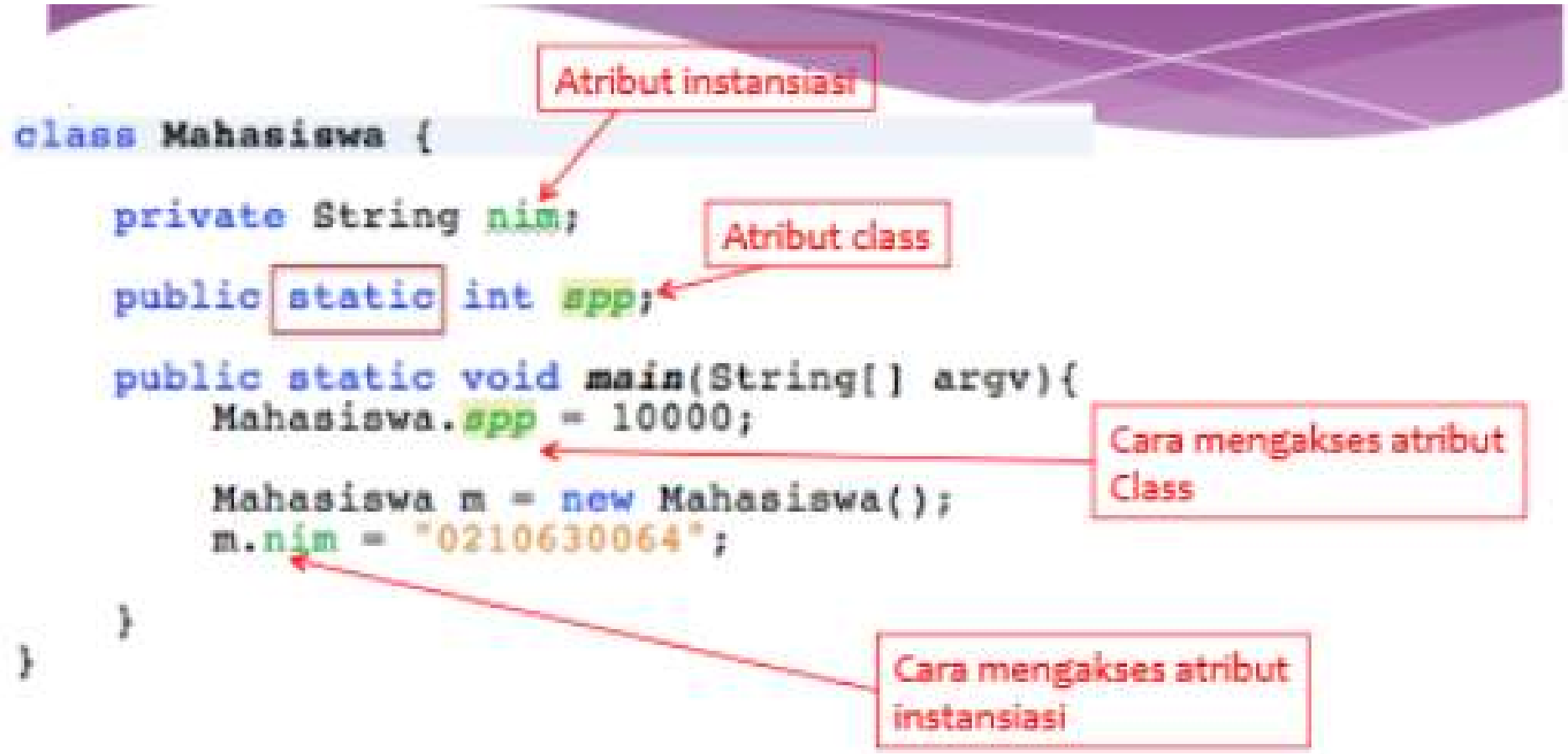
* Di Diagram Class Atribut dan method Class digambarkan dengan **garis bawah** di nama atribut&method nya



Ilustasi


Atribut instansiasi
Mahasiswa
+nim: String
+spp: int
+tampilBiodata(): void
Atribut class
instansiasi
instansiasi
Nilainya bisa beda antar objek satu dg objek lainnya
m1: Mahasiswa
+nim="0210630064"
+spp=1000
+tampilBiodata(): void
m2: Mahasiswa
+nim="0210630001"
+spp=1000
+tampilBiodata(): void
Nilainya pasti sama antar objek satu dg objek lainnya, karena niainya dimiliki oleh Class-bukan dimiliki oleh objek

Ilustrasi


Atribut instansiasi
class Mahasiswa {
private String nim;
Atribut class
public static int spp;
public static void main(String[] argv){
Mahasiswa.spp = 10000;
Cara mengakses atribut Class
Mahasiswa m = new Mahasiswa();
m.nim = "0210630064";
}
}
Cara mengakses atribut instansiasi

Setter dan Getter

- Getter adalah public method dan memiliki tipe data return, yang berfungsi untuk mendapatkan nilai dari atribut private.

- Setter adalah public method yang tidak memiliki tipe data return, yang berfungsi untuk memanipulasi nilai dari atribut private.

- Perbedaan method setter dengan getter terletak pada nilai kembalian, parameter, dan isi method-nya.

Setter dan Getter

```java
package koperasigettersetter;
public class Anggota {
    private String nama;
    private String alamat;
    private float simpanan;

    public void setNama(String nama){
        this.nama = nama;
    }
    public void setAlamat(String alamat){
        this.alamat = alamat;
    }
    public String getNama(){
        return nama;
    }
    public String getAlamat(){
        return  alamat;
    }
    public float getSimpanan(){
        return simpanan;
    }
    public void setor(float uang){
        simpanan +=uang;
    }
    public void pinjam(float uang){
        simpanan -=uang;
    }
}
```

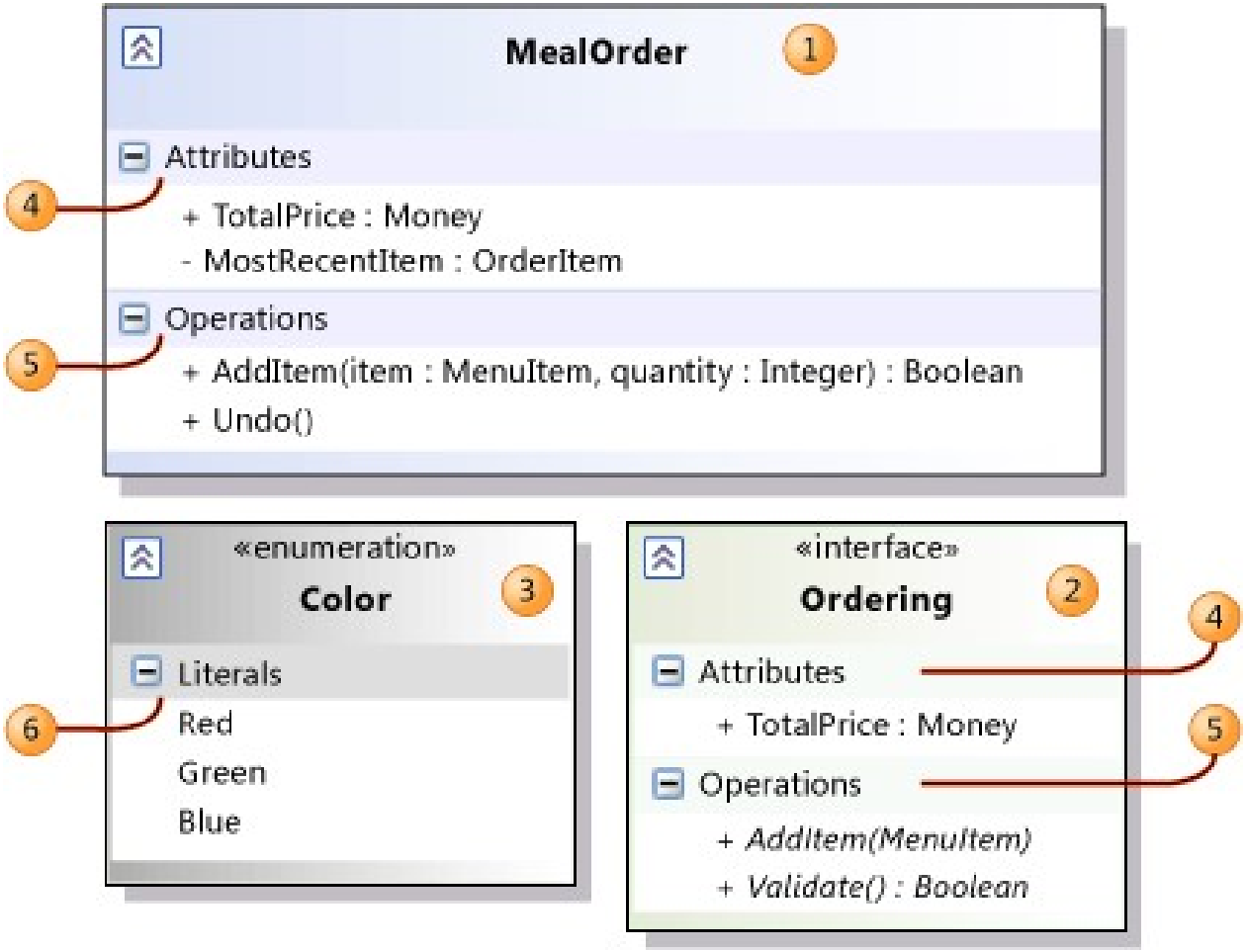
setter, getter nama dan alamat

getter simpanan

```java
package koperasigettersetter;
public class KoperasiDemo {
    public static void main(String[] args) {
        Anggota anggota1 = new Anggota();
        anggota1.setNama("Iwan Setiawan");
        anggota1.setAlamat("Jalan Sukarno Hatta no 10");
        anggota1.setor(100000);
        System.out.println("Simpanan " +anggota1.getNama()+ " : Rp "+ anggota1.getSimpanan());

        anggota1.pinjam(5000);
        System.out.println("Simpanan " +anggota1.getNama()+ " : Rp "+ anggota1.getSimpanan());
    }
}
```

Notasi Pada UML Class Diagram



Notasi Pada UML Class Diagram

- Notasi akses modifier pada UML class diagram adalah sebagai berikut:
  - Tanda plus (+) untuk public
  - Tanda pagar (#) untuk protected
  - Tanda minus (-) untuk private
  - Untuk default, maka tidak diberi notasi