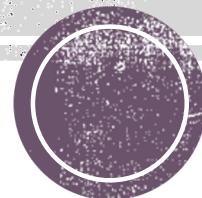


Pemrograman Berorientasi Objek :

Konsep Pemrograman Berorientasi Objek

Royana Afwani



Bahasa Pemrograman?

- Komputer bekerja seperti **switching** dan hanya **mengenali 0 dan 1**
- Manusia **tidak (paham)** berbicara dengan bahasa 0 dan 1
- Perlu bahasa pemrograman yang dapat menjadi **perantara percakapan** antara komputer dan manusia
- Bahasa pemrograman diubah ke dalam bahasa yang dipahami oleh komputer dengan menggunakan **interpreter** atau **kompiler**



Compiler or Interpreter?

1. Compiler:

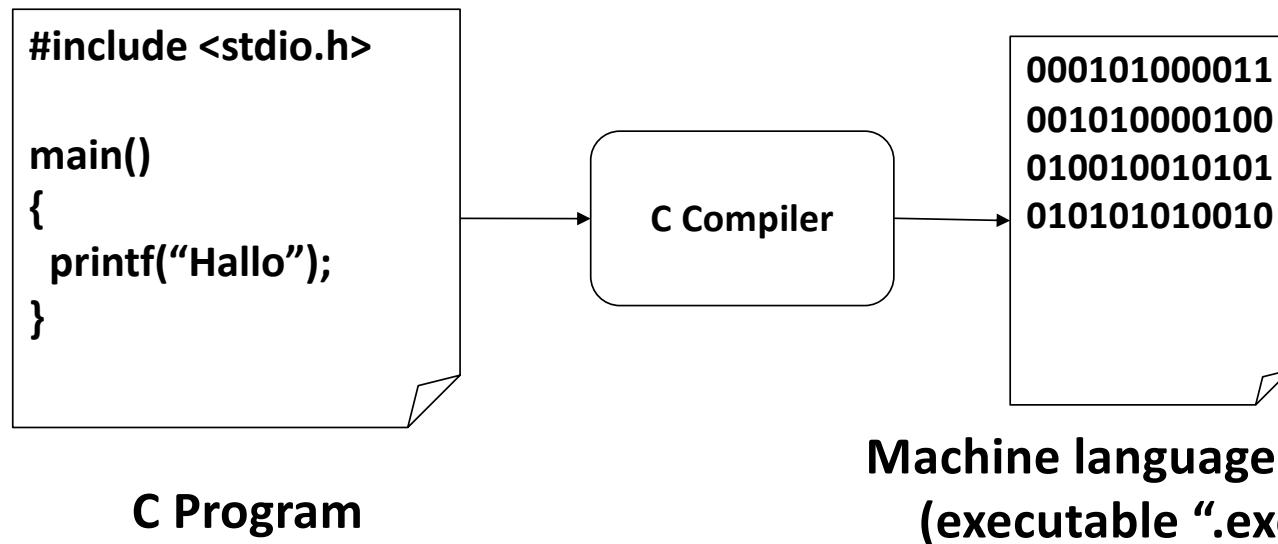
Mengkompilasi source code menjadi bentuk **file yang bisa dieksekusi**

2. Interpreter:

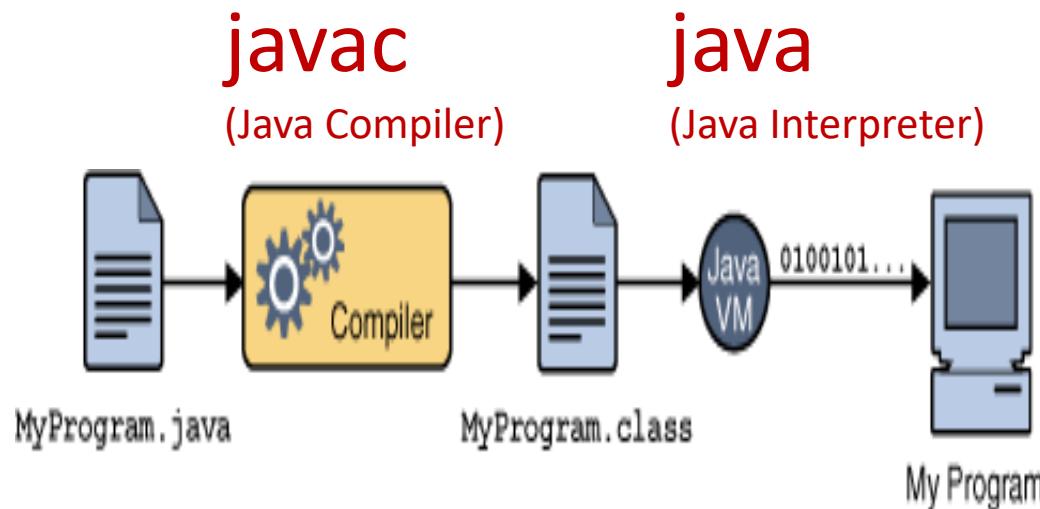
Mengkompilasi dan menjalankan source code **secara langsung**



C Language (Compiler)



Java Language (Compiler + Interpreter)

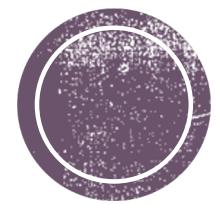


Paradigma Pemrograman

Sudut pandang dan style pemrograman berhubungan dengan bagaimana sebuah masalah diformulasikan dalam bahasa pemrograman

1. **Functional Programming**: Urutan fungsi secara **sekuensial** (Scheme, Lisp)
2. **Procedural Programming**: Pemecahan masalah berdasarkan prosedural kerja yg terkumpul dalam unit pemrograman bernama **fungsi** (C, Pascal)
3. **Object-Oriented Programming**: Koleksi object yang saling berinteraksi . **Class** adalah unit pemrograman (Java, C#, C++)



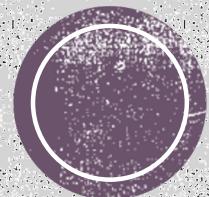


Konsep Dasar Pemrograman Berorientasi Objek

Class , Object, Method, Attribute



Objek?



Berorientasi Objek?



Attribute:

Topi, Baju, Jaket,
Tas Punggung,
Tangan, Kaki, Mata

Behavior:

Cara Jalan ke Depan
Cara Jalan Mundur
Cara Belok ke Kiri
Cara Memanjat



Berorientasi Objek?



Attribute (State):

Ban, Stir, Pedal Rem, Pedal Gas,
Warna, Tahun Produksi

Behavior:

Cara Menghidupkan Mesin
Cara Manjelaskan Mobil
Cara Memundurkan Mobil

Attribute → Variable(Member)

Behavior → Method(Fungsi)



Pengertian OOP

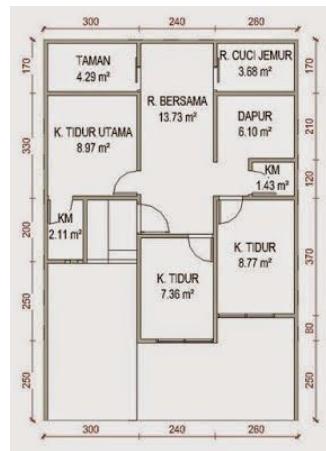
- Teknik atau cara untuk membuat program objek yang saling berinteraksi satu sama lainnya
- Objek yaitu suatu unit terkecil dari program yang mengandung data dan fungsi yang bekerja atas objek tersebut.
- Subtansi materi OOP:
 - Konsep/ilmu tentang OOP
 - Bahasa pemrograman objek: Java
 - Teknis pengoperasian tools untuk membuat program: J2SDK, IDE for Java, dll.

PERANGKAT LUNAK : BERORIENTASI OBJEK

- Perangkat lunak yang dibangun dari **kelas-kelas** dan **objek-objek** yang saling berinteraksi satu sama lainnya.
 - Kelas
Deskripsi statis dari sesuatu.
 - Objek
Sesuatu yang diciptakan (instansiasi) dari kelas.



Kelas vs Objek



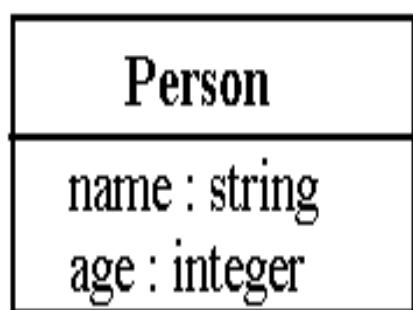
Perbedaan Class dan Object

- Class: **konsep** dan **deskripsi** dari sesuatu
 - Class mendeklarasikan **method** yang dapat digunakan (dipanggil) oleh object
- Object: **instance dari class**, bentuk (contoh) nyata dari class
 - Object memiliki sifat **independen** dan dapat digunakan untuk memanggil method
- Contoh Class dan Object:
 - Class: **mobil**
 - Object: **mobilnya pak Joko, mobilku, mobil berwarna merah**

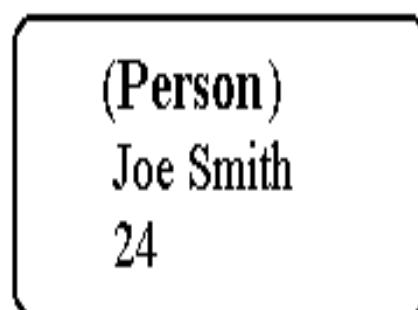


Perbedaan Class dan Object

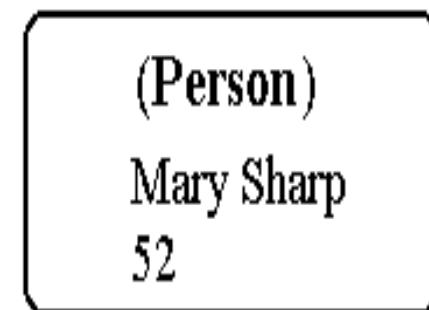
- Class seperti **cetakan kue**, dimana kue yg dihasilkan dari cetakan kue itu adalah **object**
- Warna kue bisa bermacam-macam meskipun berasal dari cetakan yang sama (**object memiliki sifat independen**)



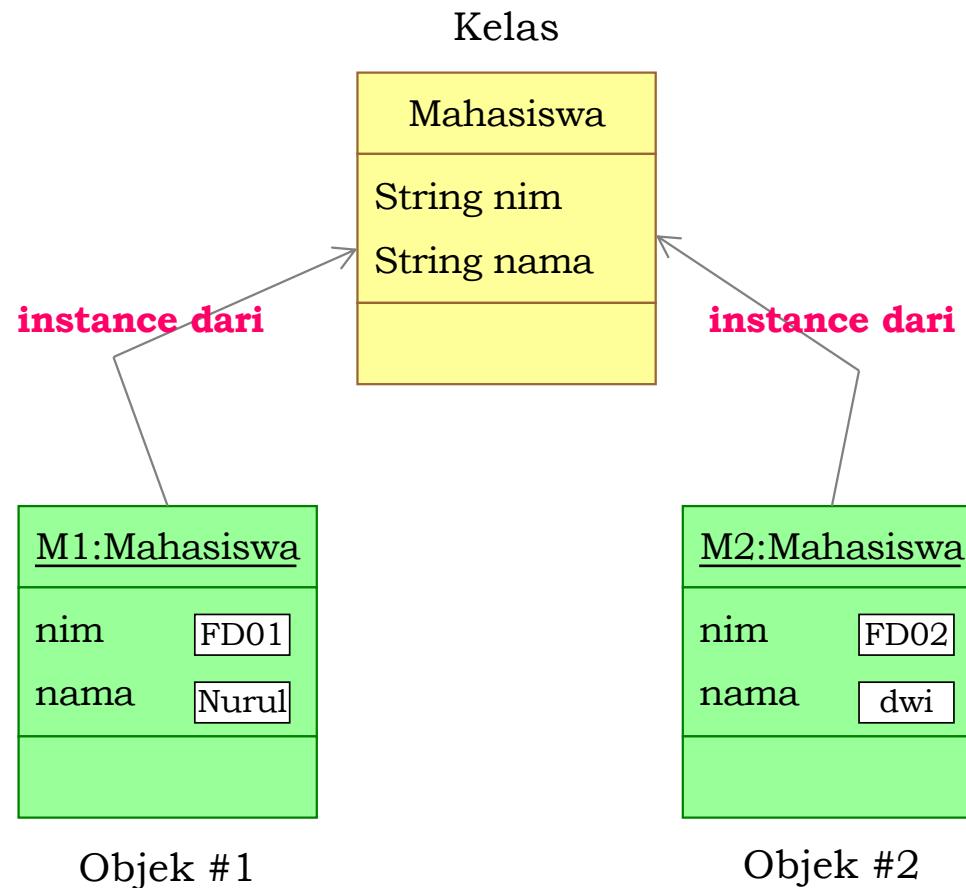
Class with Attributes



Objects with Values



Kelas dan Objek (lanjutan)



Class = Method + Variable

Class Sepeda

gir

kecepatan

tampilkan kecepatan

ubah gir

variable

method



Object = Method + Variable yg Memiliki Nilai

Object Sepedaku

`gir = 3`

`kecepatan = 10km/jam`

`tampilkan kecepatan ()
kecepatan = 10 km/jam`

`ubah gir (2)
gir = 5`

**instance
variable**

**instance
method**



Contoh Kelas dan Objek ?

**Contoh Method (Behaviour) dan
variabel (atributnya) ?**



Property Kelas / Objek

- Atribut

Data yang mendeskripsikan kelas/objek.

Pada umumnya didefinisikan dengan *access modifer private* (konsep *information hiding*).

- Data dengan tipe data dasar: **integer**, **float**, **boolean**, **char**
- **Data majemuk**: **array**, **record**, dll.
- **Kelas** lain

- Layanan / Metode (Method)

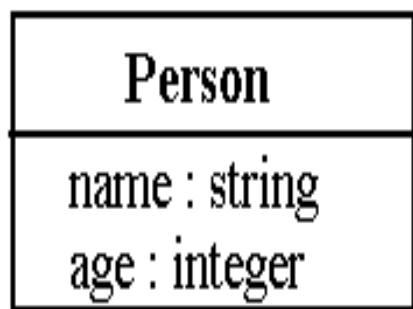
Prosedur atau **fungsi** yang mencirikan sifat atau kelakuan dari kelas/objek.

Pada umumnya didefinisikan dengan *access modifer public*.

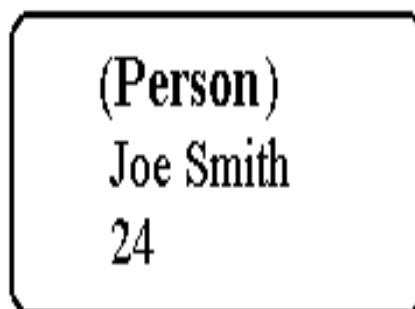
- Konstruktor: penamaan **sama** dengan nama kelasnya
- Accessor: penamaan diawali dengan kata **get**
- Mutator: penamaan diawali dengan kata **set**
- Prosedur atau fungsi tertentu: penamaan **bebas**
- Program utama: harus menggunakan nama **main()**

Attribute

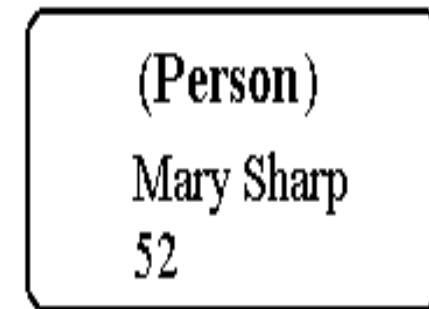
- Variable yang mengitari class, dengan nilai datanya bisa ditentukan di object
- Variable digunakan untuk menyimpan nilai yang nantinya akan digunakan pada program
- Variable memiliki jenis (tipe), nama dan nilai
- Name, age, dan weight adalah attribute (variabel) dari class Person



Class with Attributes



Objects with Values



Deklarasi Kelas

```
[public] class NamaKelas {
    // Deklarasi atribut
    ...
    // Definisi layanan/method
    <constructor>
    <accessor>
    <mutator>
    <fungsi lain>

    // Program utama
    public static void main(String arg[]) {
        ...
    }
}
```

METHOD : CONSTRUCTOR

- Fungsi yang akan dijalankan/dieksekusi secara otomatis begitu sebuah objek diciptakan.
- Diberi nama sesuai dengan nama kelasnya.

Contoh:

```
public SegiEmpat() {  
    // pernyataan  
}
```

```
public SegiEmpat(int p, int l) {  
    // pernyataan  
}
```



METHOD : SELECTOR/ACCESSOR

- Fungsi yang mempunyai kegunaan khusus untuk **mengambil nilai** dari atribut yang dimiliki sebuah objek.
- Pada umumnya menggunakan penamaan dengan format: **getNamaAtribut**

Contoh:

```
public int getPanjang() {  
    return (this.panjang);  
}
```



METHOD : MUTATOR

- Fungsi yang mempunyai kegunaan khusus untuk **memberi nilai** kepada atribut yang dimiliki sebuah objek.
- Pada umumnya menggunakan penamaan dengan format: **setNamaAtribut**

Contoh:

```
public void setPanjang(int p) {  
    this.panjang = p;  
}
```



FUNCTION main()

- Berfungsi sebagai **program utama** yang akan menjadi **titik awal eksekusi**.
- Format penulisan:

```
public static void main(String arg[]) { ... }
```

- **public**: fungsi dapat diakses dari dalam maupun luar kelas
- **static**: sama untuk semua instan dari kelas
- **void**: fungsi tidak memberikan *return value*
- **arg[]**: array String yang menjadi argumen fungsi (diambil dari lingkungan eksekusi)
- Isi fungsi main() pada umumnya adalah:
 - Penciptaan (instansiasi) objek
 - Manipulasi objek: penyalinan (copy), penggunaan



Penciptaan Objek

- Deklarasi reference atau variabel:
Mahasiswa m;
- Alokasi objek ke variabel:
m = new Mahasiswa();

Atau

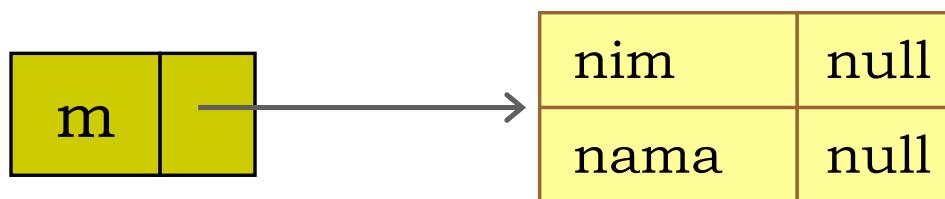
- Deklarasi dan sekaligus alokasi objek:
Mahasiswa m = new Mahasiswa();

Penciptaan Objek (lanjutan)

- Mahasiswa m; // deklarasi variabel



- m = new Mahasiswa(); // alokasi objek



PENGGUNAAN OBJEK

- Dilakukan dengan cara mengirim pesan (*message passing*) ke objek.
- Implementasinya dalam bentuk pemanggilan method yang dipunyai objek.

Contoh:

```
s.setPanjang(17);
```

nama objek

```
s.setLebar(8);
```

nama method

```
System.out.println("Luas = " + s.Luas());
```



Membuat Class, Object dan Memanggil Atribut

```
public class Mobil {  
    String warna;  
    int tahunProduksi;  
}
```

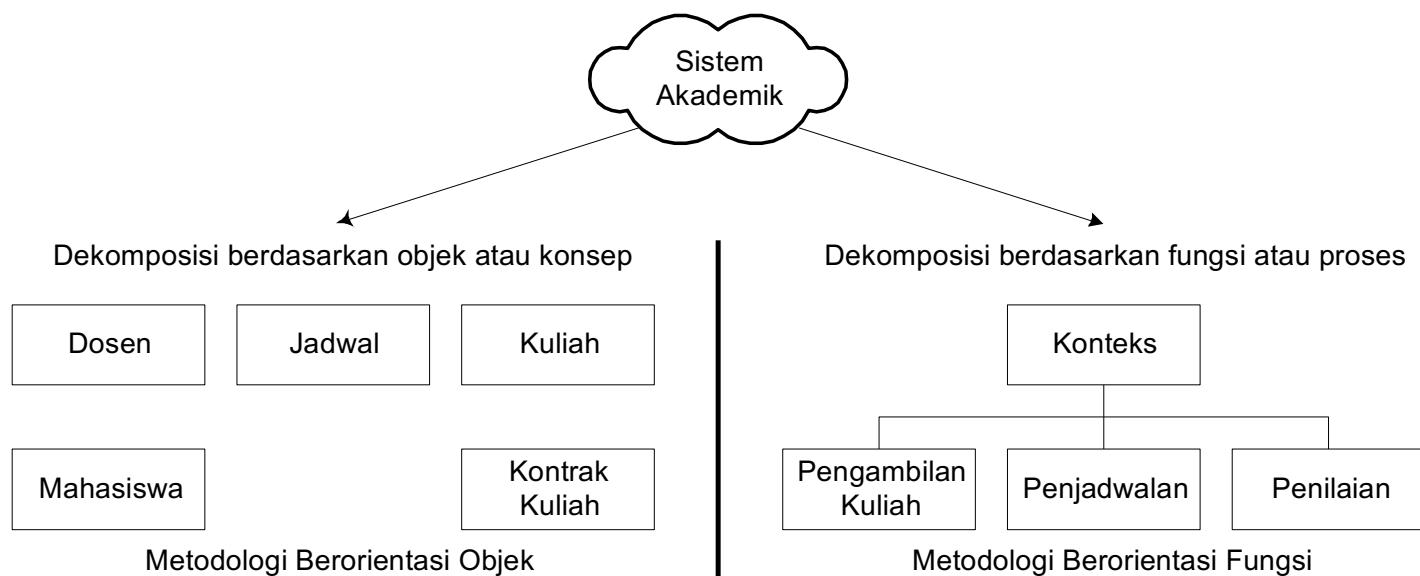
Mobil.java

```
public class MobilBeraksi{  
    public static void main(String[] args){  
        // Membuat object  
        Mobil mobilku = new Mobil();  
  
        /* memanggil atribut dan memberi nilai */  
        mobilku.warna = "Hitam";  
        mobilku.tahunProduksi = 2006;  
        System.out.println("Warna: " + mobilku.warna);  
        System.out.println("Tahun: " + mobilku.tahunProduksi);  
    }  
}
```

MobilBeraksi.java



OOP VS PROCEDURAL



Next :

- Install Java
- Standby Coding saat kuliah (Laptop)
- Submit screenshot class (coba buat kelas bebas) dengan nama kelas yaitu NIM masing-masing.

