

Pemrograman Berorientasi Objek



Pengantar OOP
INTRODUCTION TO OOP

Silabus

- › Pengantar OOP
- › Kelas dan Objek
- › Enkapsulasi
- › Hubungan Kelas
- › Warisan
- › Kelas Abstrak dan Antarmuka
- › Polimorfisme, Statis, & Koleksi
- › Pengecualian
- › UTS
- › Arsitektur MVC
- › HTML, CSS, JavaScript
- › Bahasa Jawa
- › Bahasa Inggris JDBC
- › Tabung

Komponen Pemeringkatan

- Praktikum per Minggu (12x) mulai minggu ke-2 (minggu pertama runmod): 23%
- Kuis / Tugas @ LMS: 15%
- UTS: 22%
- Aplikasi Proyek: 30%
- Dokumentasi Proyek: 5%

Aturan kursus

- Dengarkan saja kursusnya ѕ
- Kecurangan atau plagiarisme, skor maksimal adalah D
- Terlambat? Mari kita definisikan

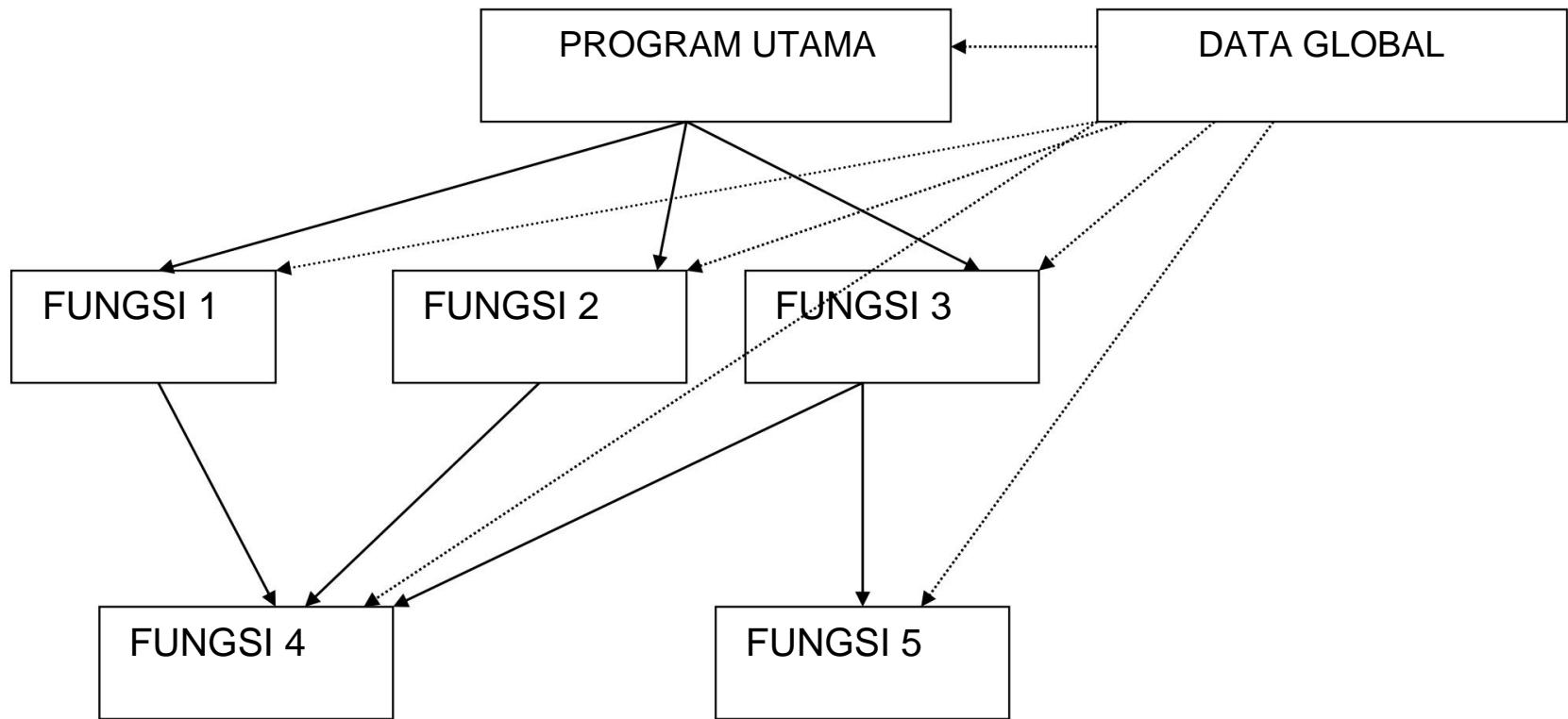
Halo semuanya!

- ...

Pengantar OOP

- › Perbedaan antara pemrograman struktural dan OOP
- › Terminologi dasar dalam OOP
- › Keuntungan dan kerugian OOP
- › Empat prinsip desain OOP
- › Kelas dan Objek
- › Variabel

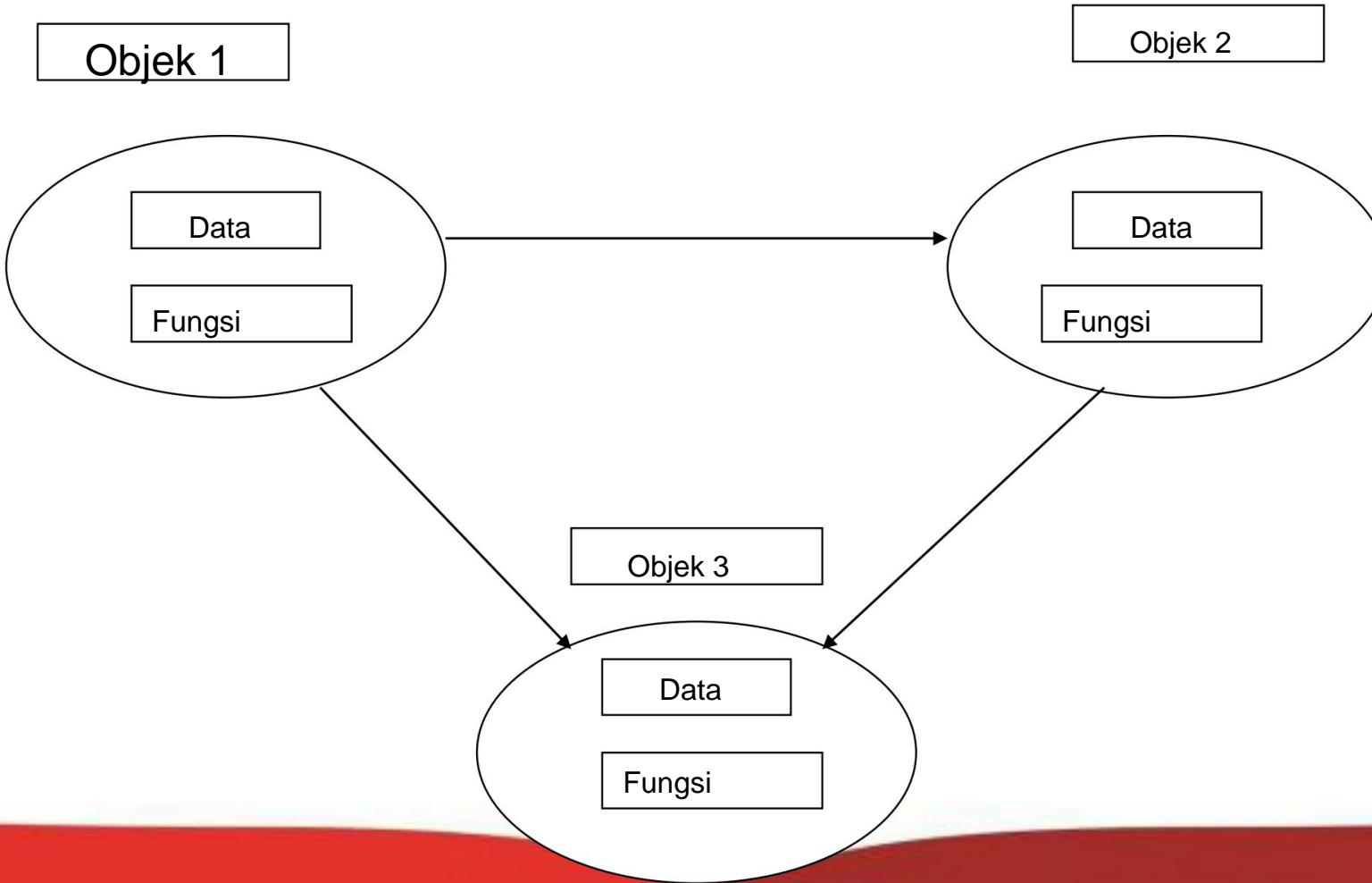
Pemrograman Terstruktur



Pemrograman Terstruktur

- › Menggunakan fungsi
- › Fungsi & program dibagi menjadi modul
- › Setiap modul memiliki data dan fungsinya sendiri yang dapat dipanggil oleh modul lain

Pemrograman Berorientasi Objek



MEMBUKA

- Objek memiliki data dan metode
- Objek dari kelas yang sama memiliki elemen data dan metode yang sama
- Objek mengirim dan menerima pesan untuk memanggil tindakan
- Dunia nyata dapat secara akurat digambarkan sebagai kumpulan objek yang berinteraksi.

Terminologi Dasar dalam OOP

- Objek: entitas dunia nyata seperti pena, laptop, mobil, tempat tidur, keyboard, mouse, kursi, transaksi, pengguna
- Kelas: sekelompok atau jenis objek yang serupa
- Metode: apa yang dapat dilakukan oleh suatu objek
 - Mobil bisa belok kanan, belok kiri, berhenti, dan lain sebagainya
- Atribut: apa yang dapat dimiliki oleh suatu objek
 - Seorang siswa dapat memiliki nama, id, nilai, dll.

Keuntungan OOP

- **Penggunaan Kembali dan Daur Ulang Kode:** Objek yang dibuat untuk Program Berorientasi Objek dapat dengan mudah digunakan kembali dalam program lain.
- **Abstraksi:** Setelah suatu Objek dibuat, pengetahuan tentang implementasinya tidak diperlukan untuk penggunaannya.
- **Penyembunyian Data:** Objek memiliki kemampuan untuk menyembunyikan bagian tertentu dari dirinya sendiri dari programmer. Ini mencegah programmer merusak nilai yang tidak seharusnya.

Keunggulan OOP (Lanjutan)

- **Manfaat Desain:** Program Berorientasi Objek memaksa desainer untuk melalui fase perencanaan yang ekstensif, yang menghasilkan desain yang lebih baik dengan lebih sedikit kekurangan.
- **Pemeliharaan Perangkat Lunak:** Program Berorientasi Objek jauh lebih mudah dimodifikasi dan dipelihara daripada Program Non-Berorientasi Objek.

Kerugian OOP

- ▶ **Ukuran:** Program Berorientasi Objek jauh lebih besar daripada program lain.
- ▶ **Upaya:** Program Berorientasi Objek memerlukan banyak pekerjaan untuk dibuat.
- ▶ **Kecepatan:** Program Berorientasi Objek lebih lambat daripada program lain, sebagian karena ukurannya.
Selain itu, OOP membutuhkan lebih banyak sumber daya sistem, sehingga memperlambat program.

Prinsip Desain OOP

- › Abstraksi
- › Enkapsulasi
- › Warisan
- › Polimorfisme

Abstraksi

- Berfokuslah hanya pada fakta-fakta penting tentang masalah yang dihadapi untuk merancang, memproduksi, dan menjelaskan sehingga dapat dengan mudah digunakan tanpa mengetahui detail cara kerjanya.

Analogi:

- Saat Anda mengendarai mobil, Anda tidak perlu tahu bagaimana bensin dan udara dicampur dan dinyalakan.
- Sebaliknya Anda hanya perlu tahu cara menggunakan kontrolnya.

Enkapsulasi

- Juga dikenal sebagai penyembunyian data
- Hanya metode objek yang dapat mengubah informasi dalam objek.

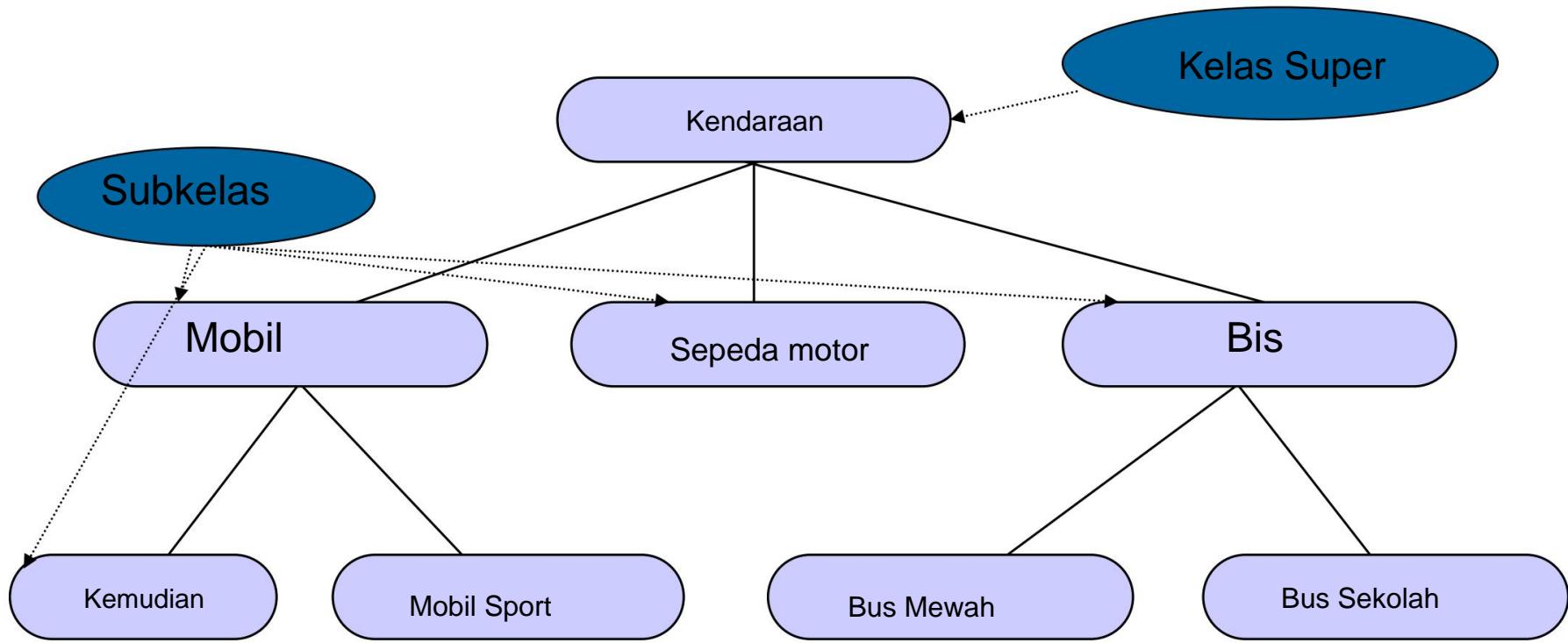
Analogi:

- Mesin ATM hanya dapat memutakhirkan akun satu orang atau objek saja.

Warisan

- Suatu cara mengatur kelas
- Kelas-kelas dengan properti yang sama dapat dikelompokkan sehingga properti umumnya hanya didefinisikan satu kali.
- Superclass – mewarisi atribut dan metode ke subclass.
- Subkelas – dapat mewarisi semua atribut & metode superkelasnya selain memiliki atribut & metode uniknya sendiri.

Jelaskan persamaan dan perbedaan sifat-sifat tersebut!



Polimorfisme

- ▶ Objek atau perilaku yang sama dapat memiliki lebih dari satu interpretasi dalam konteks yang berbeda
- ▶ Contoh:
 - Bentuknya bisa berupa segitiga, persegi, atau lingkaran
 - Cara setiap makhluk hidup bergerak berbeda-beda. Ikan bergerak dengan cara berenang, burung bergerak dengan cara terbang, dan lain sebagainya.

Pertanyaan?



KELAS

MEMBUKA

- Mewakili dunia nyata: Bayi

Baby

```
String name
boolean isMale
double weight
double decibels
int numPoops
```

Mengapa menggunakan kelas?

```
String nameAlex;  
double weightAlex;  
// little baby alex  
String nameDavid;  
double weightDavid;  
// little baby david  
String nameDavid2;  
double weightDavid2;
```



David2?
Terrible 😞

500 Babies? That Sucks!

Lanjutan.



Baby1



Baby2



Baby3

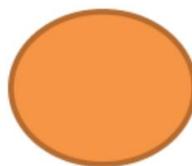


Baby4

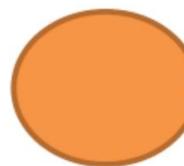
496
more
Babies

...

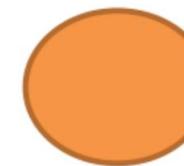
Lanjutan.



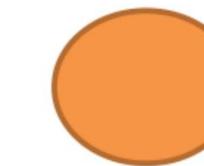
Nurse1



Nurse2



Nurse3

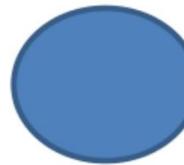


Nurse4

More nurses...



Baby1



Baby2



Baby3



Baby4

496 more
Babies ...

Membuat Kelas

```
public class Baby {
```

fields

methods

}

Catatan

- Nama kelas menggunakan huruf kapital
- 1 kelas = 1 berkas
- Memiliki metode utama berarti kelas dapat dijalankan

Kelas menulis

```
public class Baby {  
    String name;  
    boolean isMale;  
    double weight;  
    double decibels;  
    int numPoops = 0;  
  
    void poop() {  
        numPoops += 1;  
        System.out.println("Dear mother, "+  
            "I have pooped. Ready the diaper.");  
    }  
}
```

Kelas Desain

Mulailah dari objek konkret lalu generalisasikan

- Hal-hal yang **diketahui** objek
- Hal-hal yang **dilakukan** objek

Contoh

cartContents

knows

addToCart()
removeFromCart()
checkOut()

does

Alarm

alarmTime
alarmMode

setAlarmTime()
getAlarmTime()
isAlarmSet()
snooze()

knows

does

Button

label
color

setColor()
setLabel()
dePress()
unDepress()

knows

does

Lanjutan.

**instance
variables**

(state)

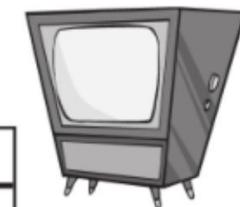
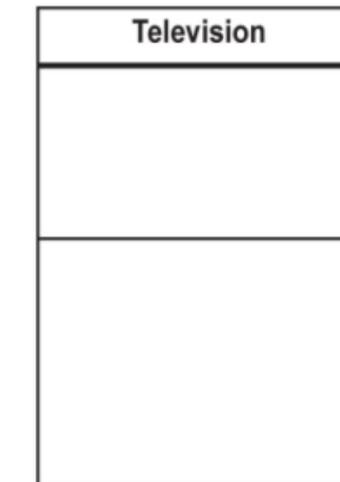
methods

(behavior)



knows

does



**instance
variables**

methods

Lanjutan.

Name
(Identifier)
Variables
(Static attributes)
Methods
(Dynamic behaviors)

| Student |
|--------------|
| name |
| grade |
| getName() |
| printGrade() |

| Circle |
|-------------|
| radius |
| color |
| getRadius() |
| getArea() |

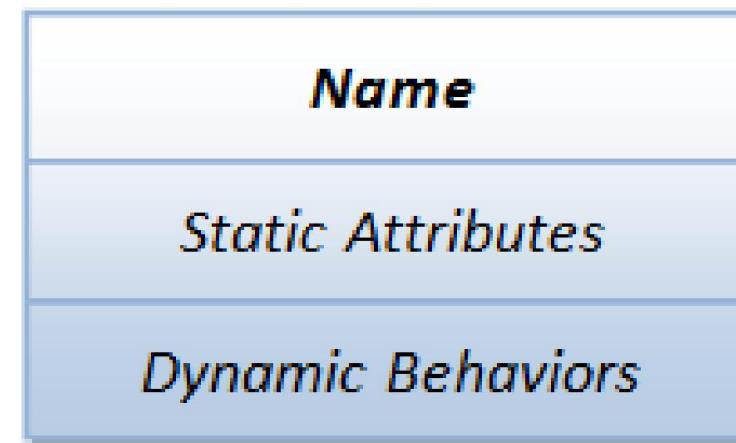
| SoccerPlayer |
|--------------|
| name |
| number |
| xLocation |
| yLocation |
| run() |
| jump() |
| kickBall() |

| Car |
|--------------|
| plateNumber |
| xLocation |
| yLocation |
| speed |
| move() |
| park() |
| accelerate() |

Examples of classes

Ringkasan

- › *kelas adalah definisi dari objek yang sejenis*
- › *Nama* (atau identitas): mengidentifikasi kelas.
- › *Variabel* (atau atribut, status, bidang): berisi *atribut statis* kelas.
- › *Metode* (atau perilaku, fungsi, operasi): berisi *perilaku dinamis* kelas.

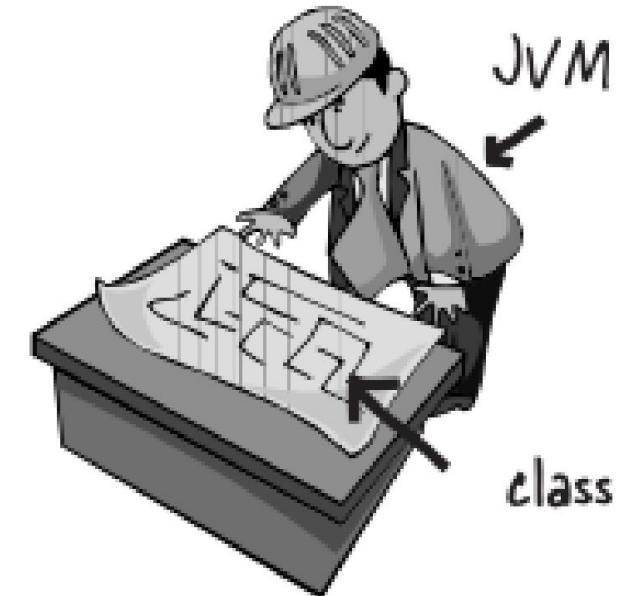
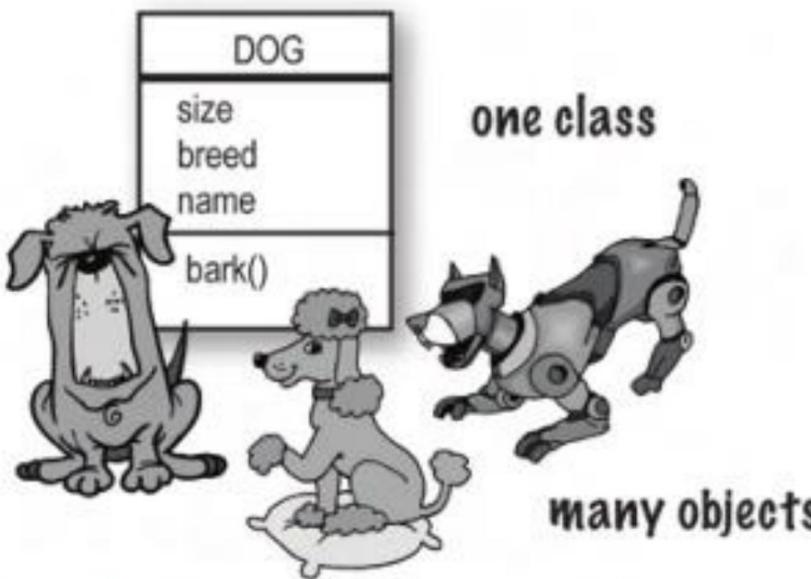


A class is a 3-compartment box

OBYEK

Kelas bukanlah objek

- › Satu kelas, banyak objek
- › Kelas yang digunakan untuk membuat objek (menggunakan kata kunci baru)
- › Kelas adalah “cetak biru” atau “resep” untuk objek



Menjalankan objek Anda

1. Tulis kelas Anda
2. Tulis kelas penguji (driver)
3. Pada penguji Anda, buatlah objek dan mengakses variabel dan metode objek

Menulis kelas

```
kelas Anjing{
```

```
    int ukuran;
```

```
    Jenis tali;
```

```
    Nama string;
```

```
    kekosongan kulit kayu(){
```

```
        System.out.println("Wah! Wah!");
```

```
}
```

```
}
```



Tulis kelas pengujii/pengemudi

kelas DogTest {

 publik statis void utama (String[] args){

 // tes akan ada di sini

}

}

Tes tulis

```
kelas DogTest {
```

```
    publik statis void utama (String[] args){
```

```
        Anjing d = new Dog(); // membuat objek (istilah: membuat instance)
```

```
        d.size = 40; // atur ukuran anjing
```

```
        d.bark(); //memanggil metode bark
```

```
}
```

```
}
```

Dua penggunaan utama (kelas penguji/pengemudi)

- Uji coba Anda yang sebenarnya
 - Dimungkinkan untuk memiliki main di setiap kelas untuk menguji kelas itu sendiri

- Untuk meluncurkan/memulai Aplikasi Java Anda
 - utama adalah titik masuk

Karakteristik OOP

1. Segala sesuatu adalah sebuah objek
2. Program adalah sekumpulan objek yang saling memberi tahu apa yang harus dilakukan dengan mengirimkan pesan
3. Setiap objek memiliki memori sendiri yang terdiri dari objek lain
4. Setiap objek memiliki tipe
5. Semua objek dengan tipe tertentu dapat menerima pesan yang sama

Definisi yang lebih ringkas

Suatu Objek memiliki status, perilaku dan identitas

VARIABEL

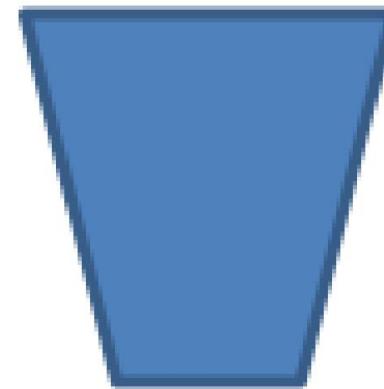
2 Aturan untuk variabel

1. Variabel harus memiliki tipe
2. Variabel harus memiliki nama

```
int count;  
      ↑  
  type  
      ↑  
    name
```

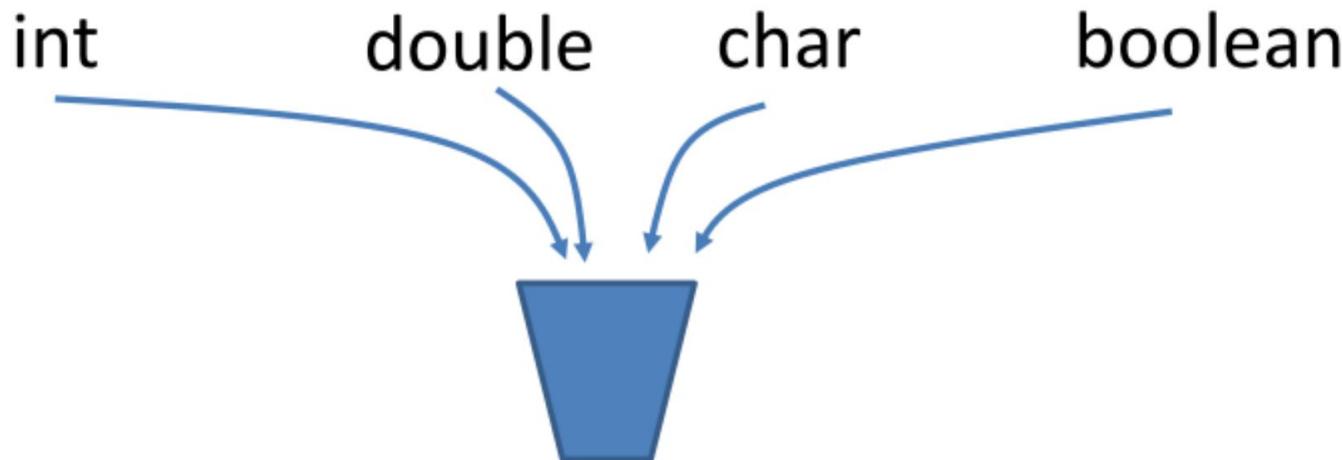
Variabel apa yang dimaksud?

- › Variabel hanyalah sebuah cangkir
- › Sebuah wadah. Itu menampung sesuatu



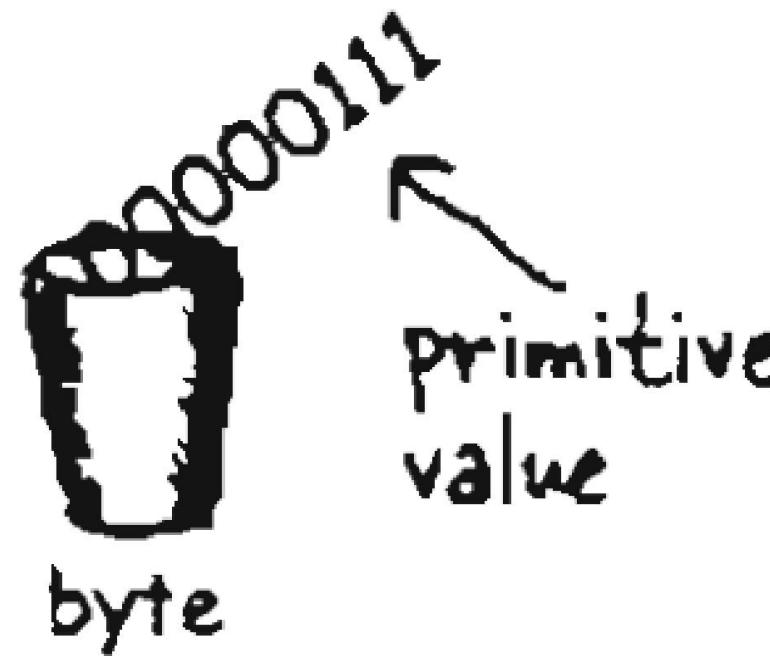
Variabel tipe primitif int, long,

- › double, boolean, char, short, byte, float
- › Nilai sebenarnya disimpan dalam variabel
- › Primitif cukup kecil sehingga bisa dimasukkan ke dalam cangkir



Contoh

byte x = 7;



Variabel tipe referensi

- › Semua kecuali tipe primitif: objek, array
- › Objek terlalu besar untuk dimasukkan ke dalam variabel (cangkir)



Lanjutan.

- Sebenarnya tidak ada yang namanya variabel objek
- Hanya ada variabel **referensi objek**
- Variabel referensi objek menyimpan bit yang mewakili cara mengakses objek
- Ia tidak menyimpan objek itu sendiri, tetapi menyimpan sesuatu seperti pointer (tetapi bukan pointer seperti C)

Anda memanipulasi objek dengan referensi

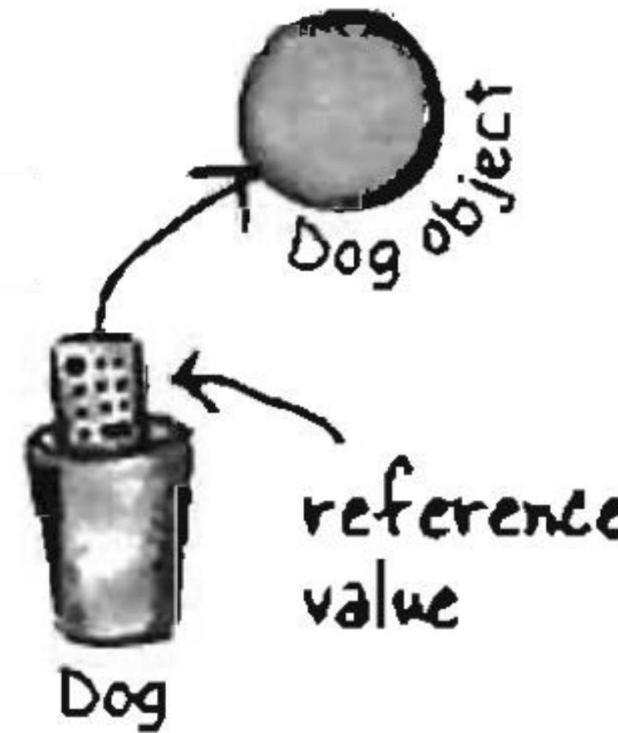
- Meskipun Anda memperlakukan segala sesuatu sebagai sebuah objek, pengenal yang Anda manipulasi sebenarnya adalah sebuah "referensi" ke sebuah objek.
- Anda mungkin membayangkan sebuah televisi (objek) dan pengendali jarak jauh (referensi).
- Selama Anda memegang referensi ini, Anda memiliki koneksi ke televisi, tetapi saat seseorang berkata, "Ganti saluran" atau "Kecilkan volume," yang Anda manipulasi adalah referensinya, yang pada gilirannya memodifikasi objek.
- Jika Anda ingin bergerak keliling ruangan dan tetap dapat mengontrol televisi, bawalah remote/referensi, bukan televisinya.

Referensi sebagai jarak jauh



Contoh

```
Anjing myDog = new Anjing();
```



3 Langkah Pembuatan

Objek 1. Mendeklarasikan variabel referensi

–Dog myDog; 2.

Membuat objek –new Dog()

3.Tautkan objek ke referensi –

Dog myDog = new Dog();

Tempat program berada

1. Register (di dalam prosesor)
2. Tumpukan (dalam RAM tetapi memiliki dukungan langsung dari prosesor)
3. Heap (juga dalam RAM, sangat fleksibel)
4. Penyimpanan konstan (ROM)

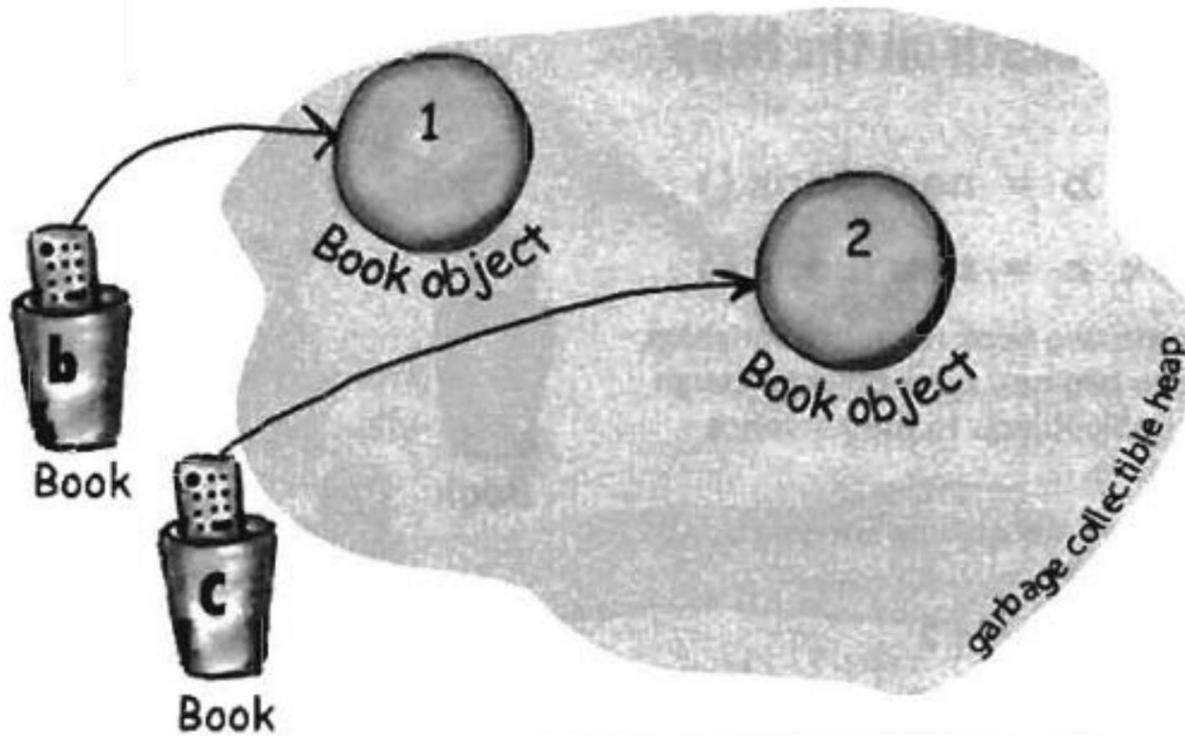
3 hal yang perlu diingat

- › **Variabel instan** dan **objek** ada di **heap**
- › **Variabel lokal** ada di **tumpukan**

Kehidupan dan kematian suatu objek

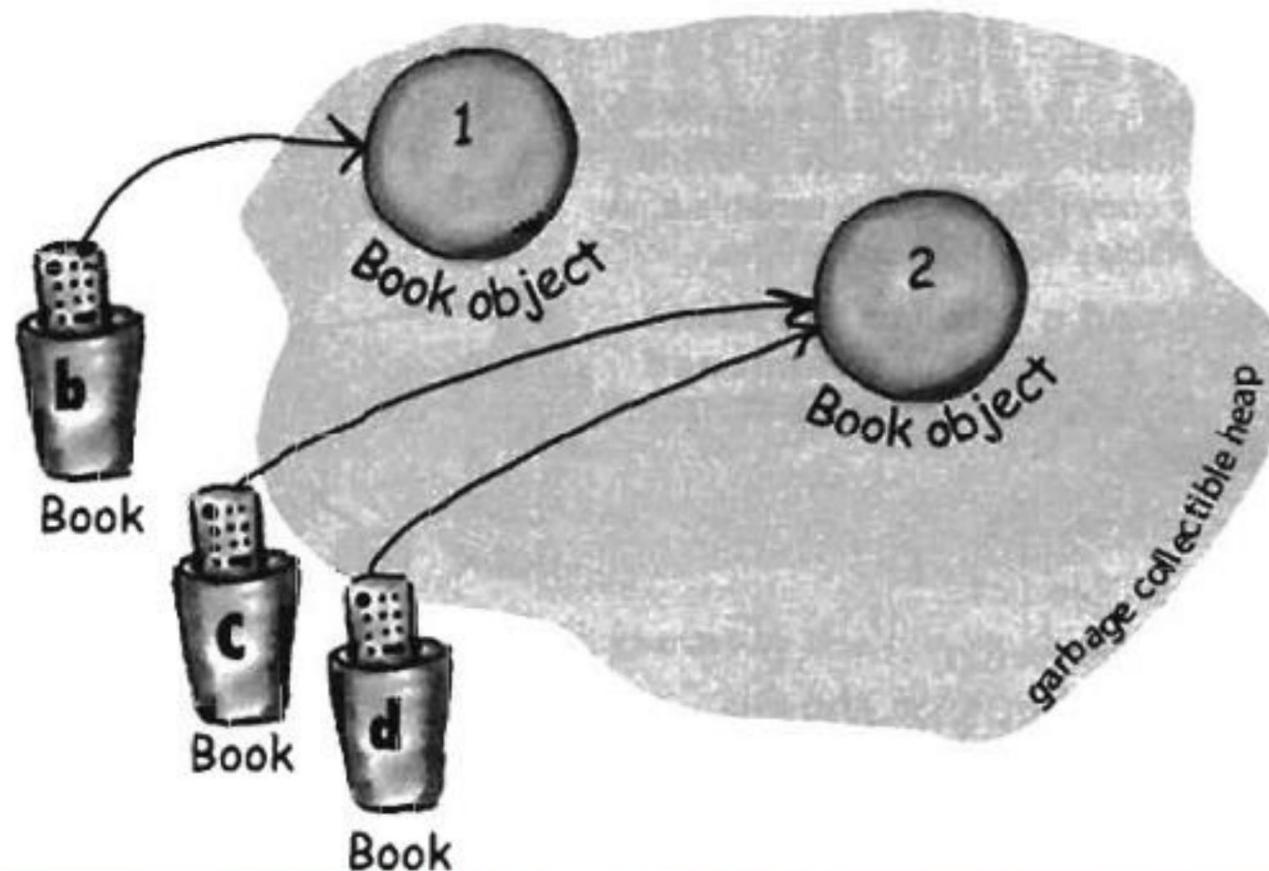
Buku b = new Buku();

Buku c = new Buku();



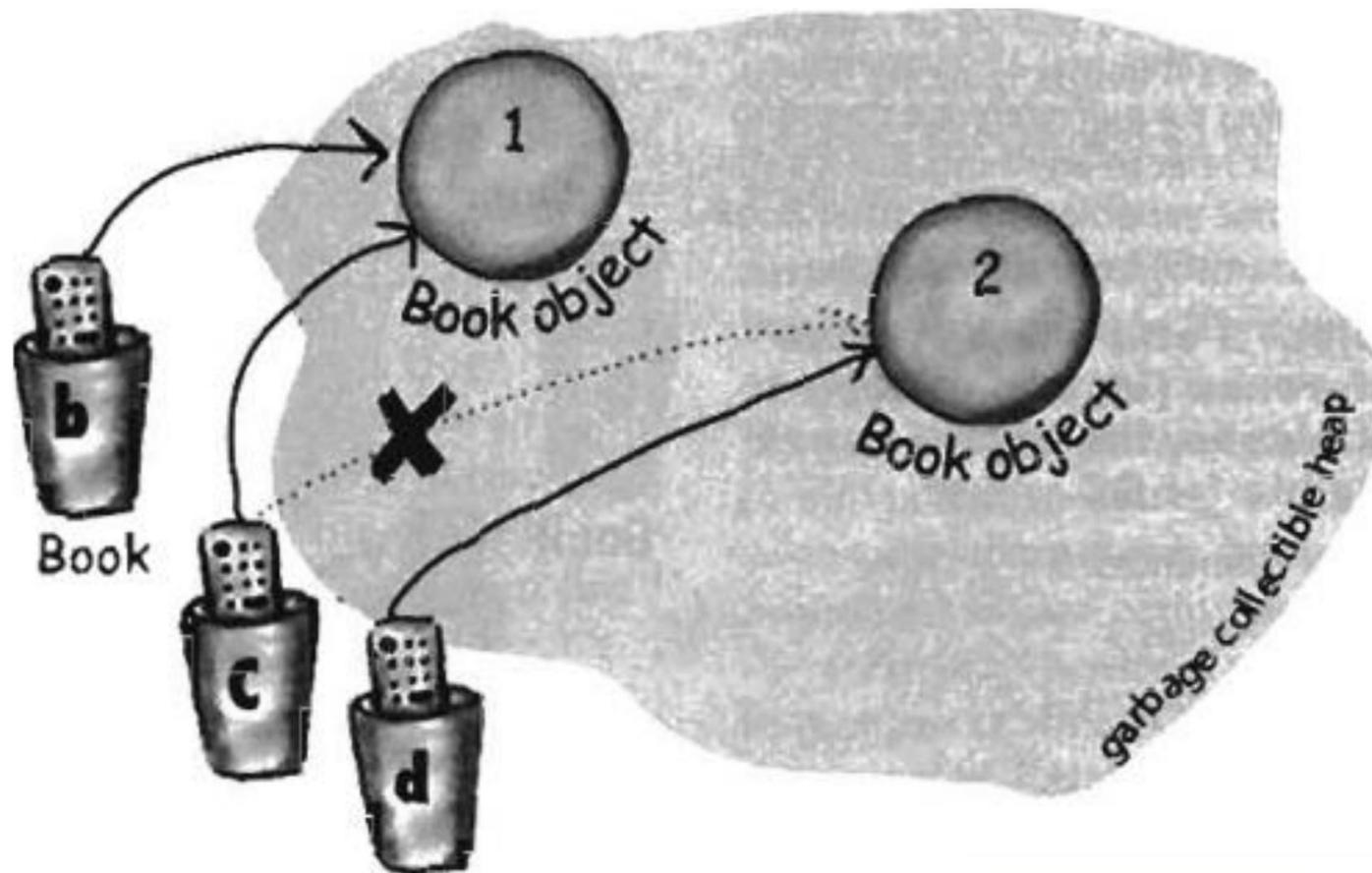
Lanjutan.

Buku d = c;



Lanjutan.

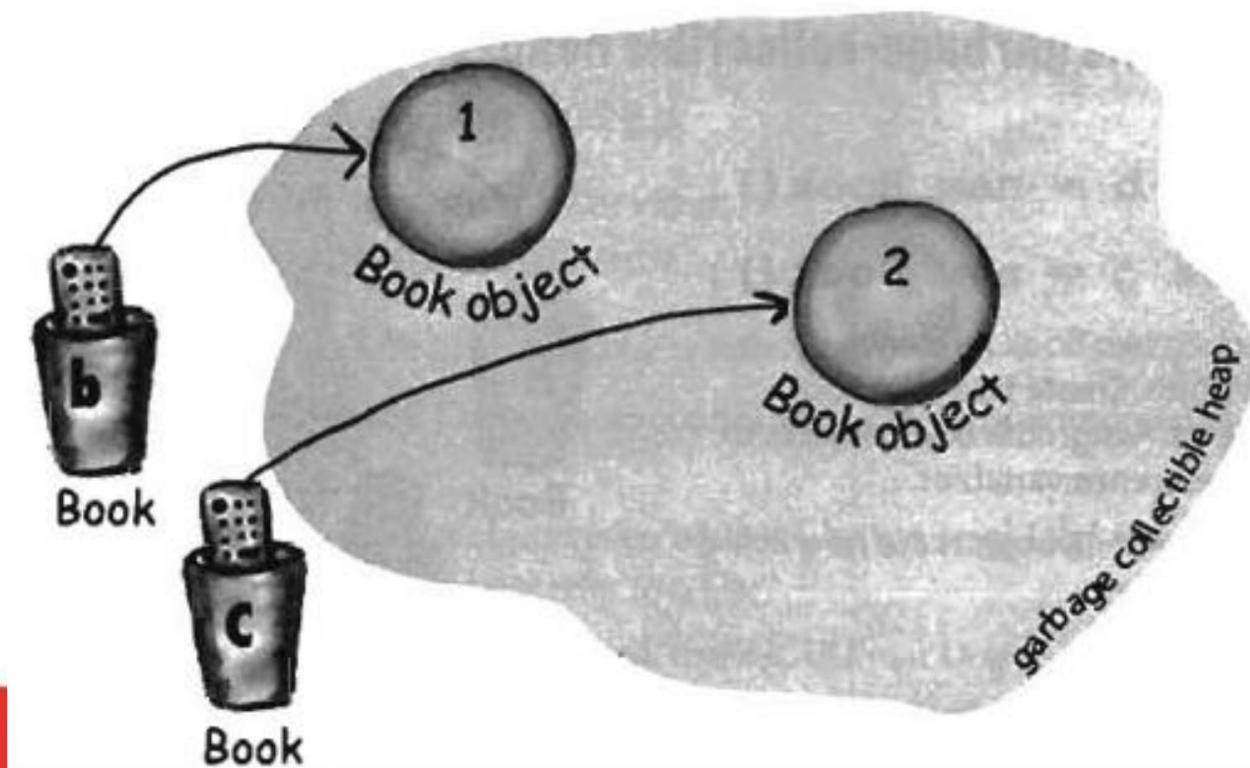
Bahasa Indonesia: c = b;



Lanjutan.

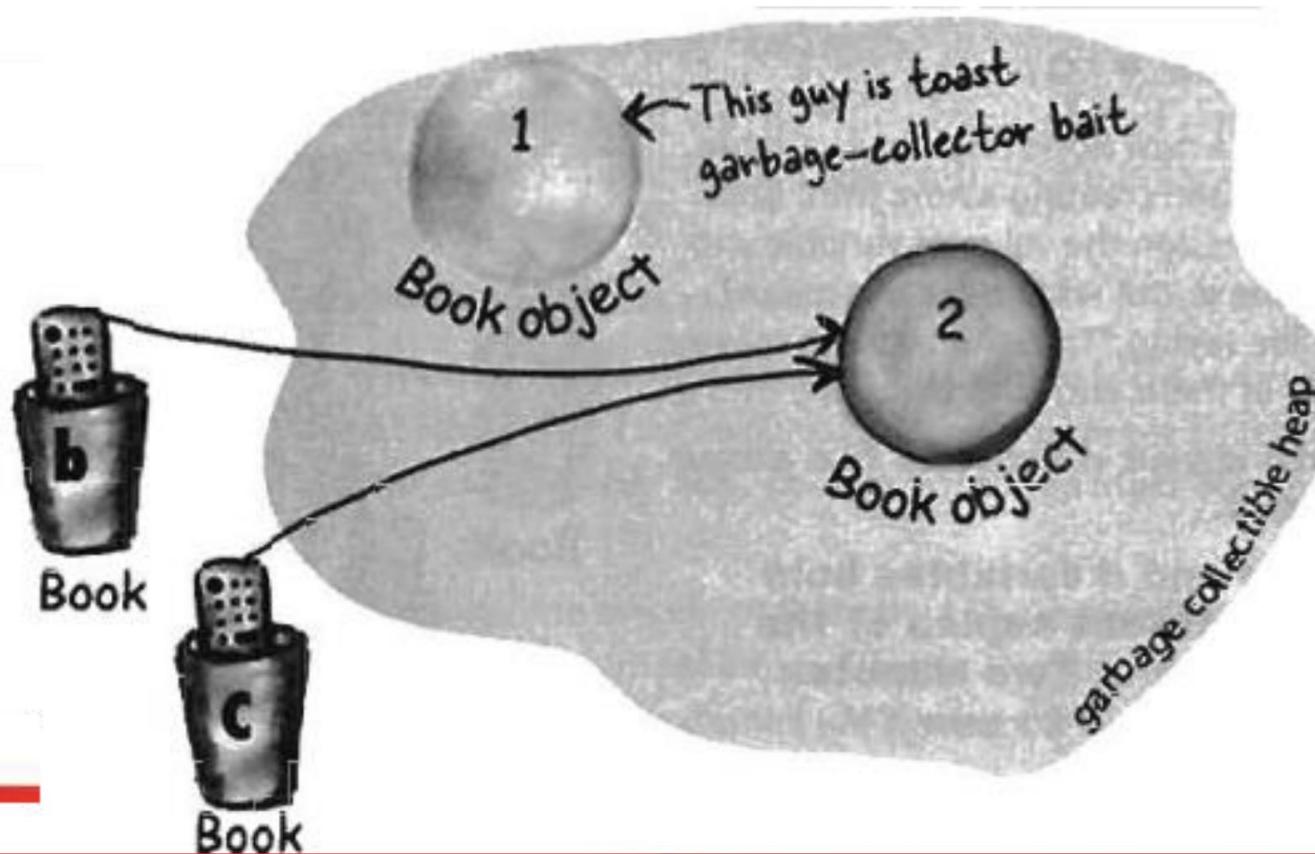
Buku b = new Buku();

Buku c = new Buku();



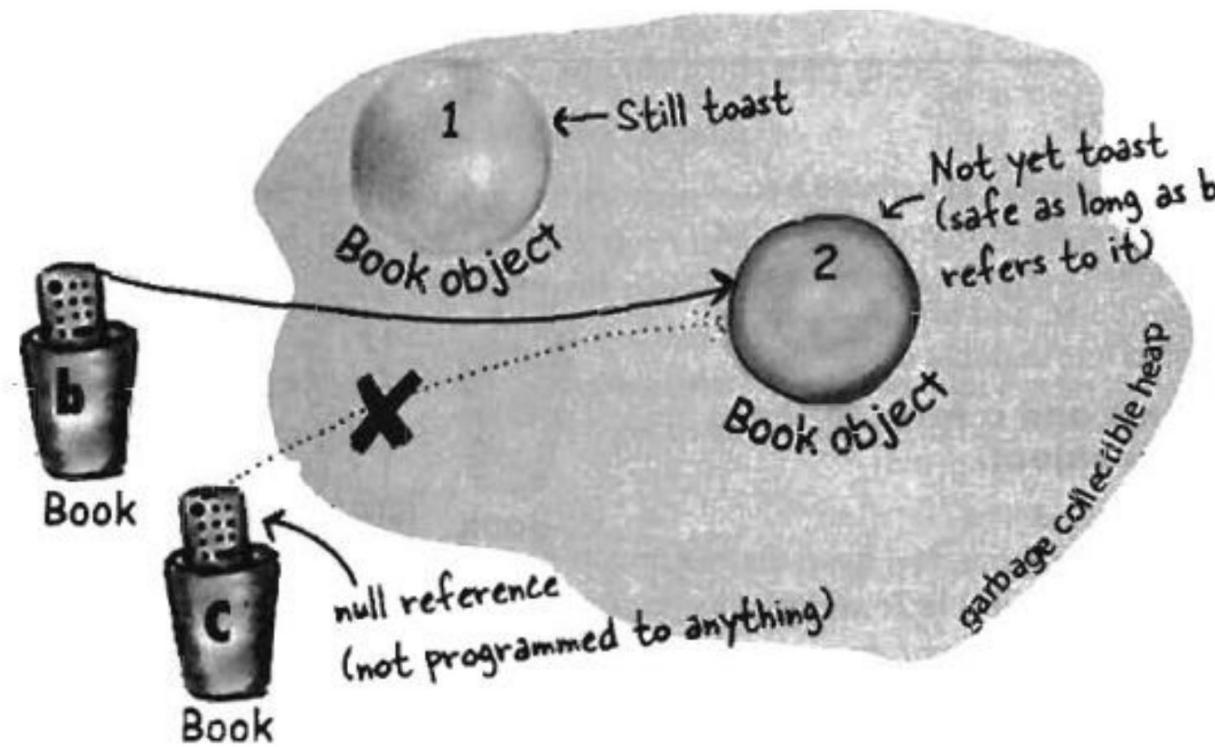
Lanjutan.

Bahasa Indonesia: $b = c;$



Lanjutan.

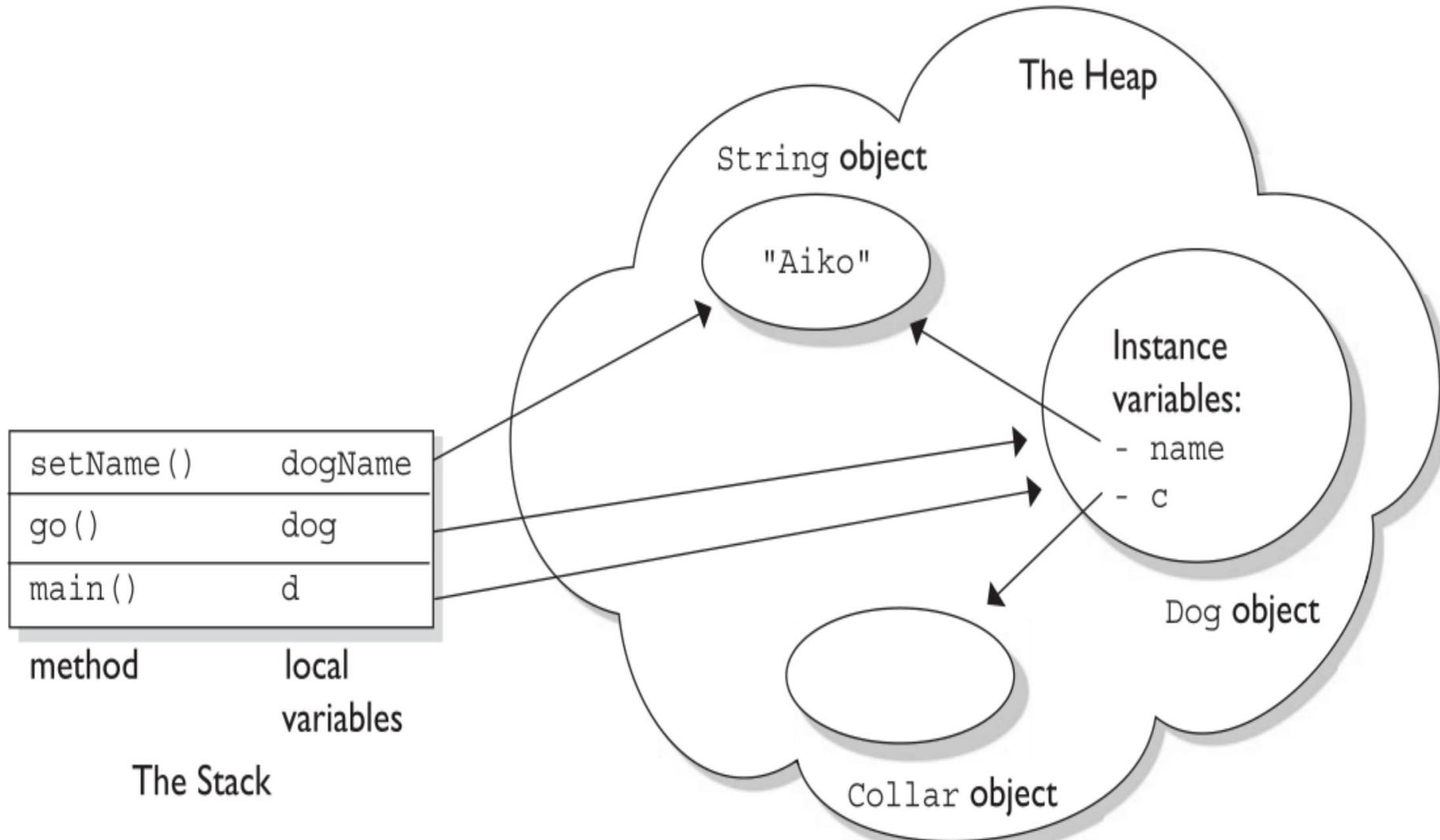
c = nol;



Contoh kompleks

```
1. class Collar { }
2.
3. class Dog {
4.     Collar c;           // instance variable
5.     String name;        // instance variable
6.
7.     public static void main(String [] args) {
8.
9.         Dog d;           // local variable: d
10.        d = new Dog();
11.        d.go(d);
12.    }
13.    void go(Dog dog) {           // local variable: dog
14.        c = new Collar();
15.        dog.setName("Aiko");
16.    }
17.    void setName(String dogName) { // local var: dogName
18.        name = dogName;
19.        // do more stuff
20.    }
21. }
```

Ilustrasi (membuat animasi)



Pertanyaan?





TERIMA KASIH