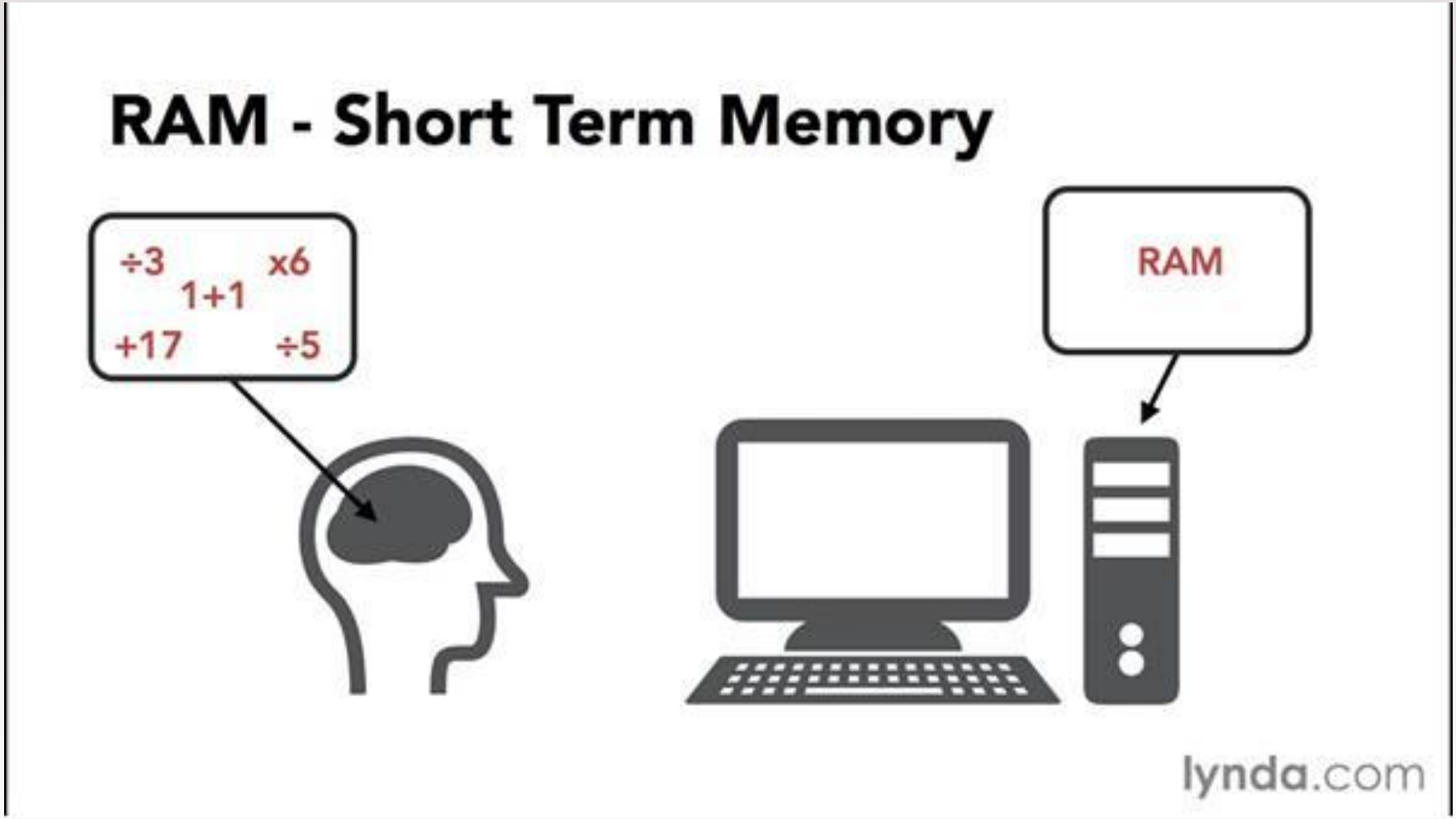
Pengantar Struktur Data I (pointer java)

README.MD

Pengantar untuk struktur data, karena ada materi seperti pointer yang harusnya dipahami(kadang tidak dipelajari atau tidak dipahami) dulu sebelum masuk ke struktur data.

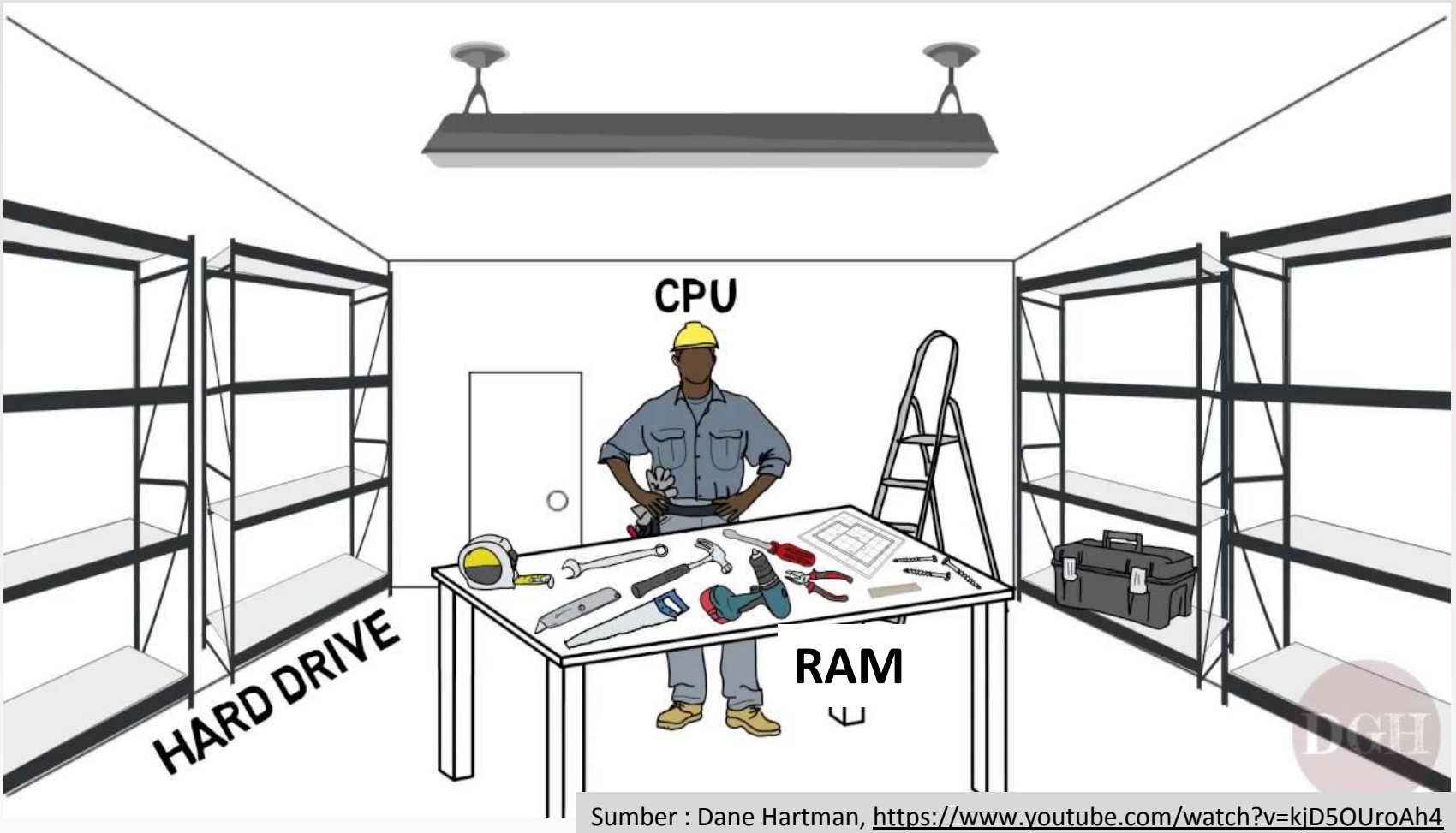
RANDOM ACCESS MEMORY

Analogi 1



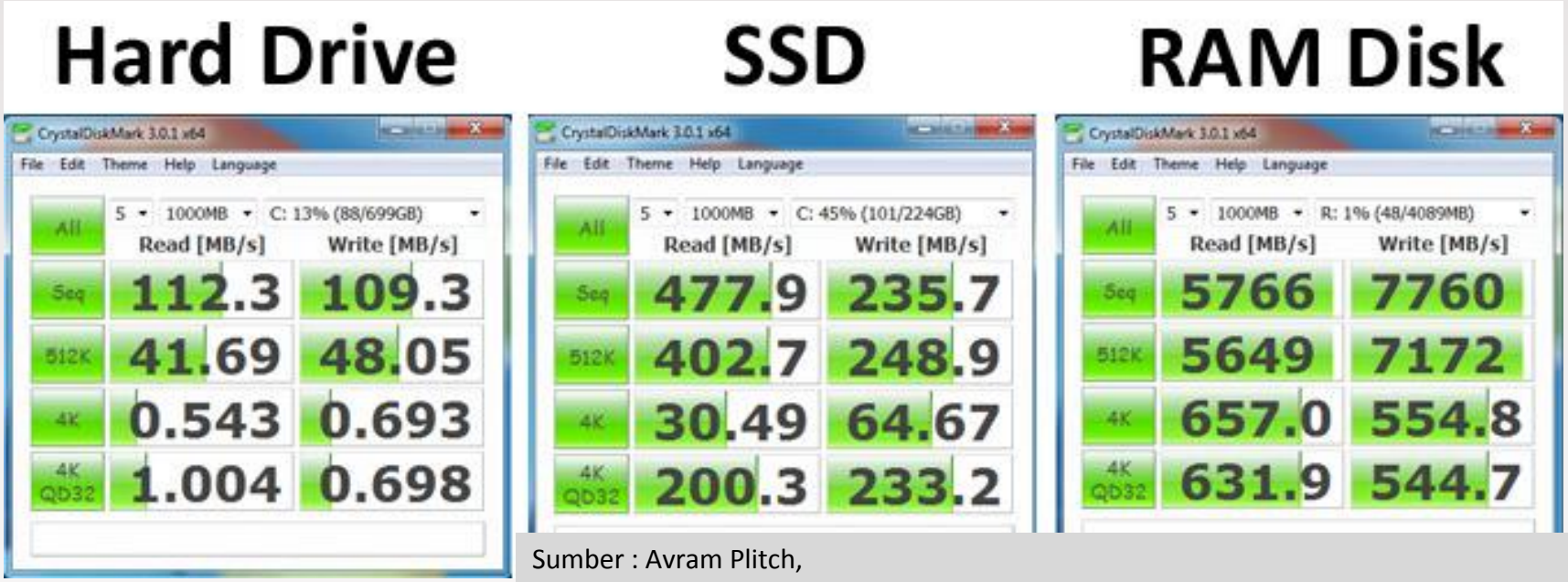
Sumber : Nick Brazzi, https://www.lynda.com/Mac-OS-Server-tutorials/Understanding-how-processor-RAM-hard-drive-performance-affect-speed/191499/366317-4.html

Analogi 2



Sumber : Dane Hartman, https://www.youtube.com/watch?v=kjD5OUroAh4

Analogi 3



Sumber : Avram Plitch, https://www.laptopmag.com/articles/faster-than-an-ssd-how-to-turn-extra-memory-into-a-ram-disk

Pemrosesan Data

- Pemrosesan data dilakukan di CPU yang mempunyai kecepatan tinggi.
- Kecepatan HDD sangat lambat, apalagi dibanding CPU.
- Agar tidak terjadi *bottleneck*, data yang akan diproses ataupun diprediksi akan diproses “mengantri” di RAM.

Struktur Data

- Teknik bagaimana menyimpan data **sesuai dengan kebutuhan.** Umumnya bagaimana menyimpan di RAM.
- Karena di setiap Teknik penyimanan data pasti ada kelebihan dan kekurangan (tradeoff) dari segi **memori** dan **waktu.** Dan kita harus memilih mana Teknik yang tepat untuk program yang sesuai.
- Contoh, program yang sama menggunakan struktur data yang berbeda akan menggunakan **memori** dan **waktu** yang berbeda secara signifikan, terutama jika data dalam jumlah besar.

Bagaimana Data Disimpan dalam RAM?

- Seperti yang kita tahu, bentuk asli data adalah biner atau bit(1 dan 0)
- 1 byte = 8 bit
- 1 kilobyte = 1024 byte
- 1 megabyte = 1024 kilobyte
- 1 gigabyte = 1024 megabyte

… dan seterusnya

Bagaimana Data Disimpan dalam RAM?

Sumber : Wikipedia, https://en.wikipedia.org/wiki/Apple_II_series

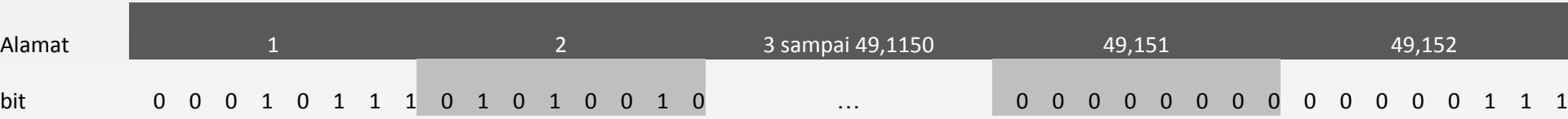
- Misal di computer Apple II dengan arsitektur 16 bit, RAM berukuran 48 kilobyte
- Kita ingin menyimpan integer yang pada saat itu berukuran 2 byte

Bagaimana Data Disimpan dalam RAM?

- Data di RAM disimpan dalam satuan byte.
- 48 kilobyte = 48 * 1024 byte = 49,152 byte
- Berarti RAM tersemut mempunyai alamat dari 1 sampai alamat ke 49,152 untuk menyimpan data.

Simulasi

Kita ingin menyimpan angka 5970, Representasi biner nya :
1011101010010
Representasi biner diatas berukuran 13 bit, Sehingga harus di pas kan menjadi 16 bit (16 bit = 2 byte (ukuran integer))
0001011101010010
Kemudian kita pecah menjadi 8 bit agar dapat disimpan di RAM
00010111 01010010
Misal integer tersebut kita simpan di alamat 1
Maka integer tersebut akan memakai 2 tempat, yaitu alamat 1 dan 2 karena integer berukuran 2 byte



Kemudian kita mencoba menyimpan angka 7 di alamat 49,151
Berapapun angkanya, ukuran nya tetap 2 byte karena tipe datanya integer.
integer tersebut akan memakai 2 tempat, yaitu alamat 49,151 dan 49,152 karena integer berukuran 2 byte

Keadaan Sebenarnya

- Dalam keadaan sebenarnya, kita tidak sewenang-wenang menentukan alamat untuk data/objek baru.
- Anggap saja sebagai abstraksi program meminta alamat ke memory manager, lalu memory manager memberi alamat nya
- Dan jika variable tersebut sudah tidak terpakai, memory manager akan automatis menghapus data di memori tersebut.

Tambahan

Strukr data juga berkaitan dengan cache, memory berukuran (sangat) kecil di CPU yang memiliki access teime (sangat) cepat

POINTER (JAVA)

Pointer

- Pointer adalah variable yang menyimpan **alamat** di memori dari sebuah data.
- Di computer umumnya alamat memori ditampilkan dalam bentuk hexadecimal (bilangan basis 16)

Keyword “new”

- Fungsi new adalah fungsi untuk memesan alamat untuk suatu data, sehingga data bisa disimpan di alamat tersebut
- Nilai kembalian (return value) dari fungsi new = alamat yang telah dipesan

Tipe Data

- Primitif :
  - int : long int, … long long int
  - float : float, double, … long long double
  - char
  - Boolean
  - Dan yang lainnya
- Non-Primitif :
  - String
  - Array
  - Class
  - Dan yang lainnya

Keyword “new”

- Untuk tipe data non-primitif(pengecualian : String) kita harus mendeklarasikan sebagai berikut agar bisa dipakai :
  - `Mahasiswa m1 = new Mahasiswa();`
  - `int arr[] = new int[10];`
- Atau bisa juga seperti ini :
  - ```
    Mahasiswa m1;
    m1 = new Mahasiswa();
    ```
  - ```
    Int arr[]
    arr = new int[3];
    ```

Keyword “new”

- Jika tanpa menggunakan keyword new (pengecualian : string), apakah data bisa digunakan (dalam Bahasa java)? Misal :
  - `Mahasiswa m1;` (baris 1)
  - `m1.setNama(“Ahmad”);` (baris 2)
- Jawaban : tidak bisa, karena perintah di baris 1 hanya membuat variabel pointer (m1) yang nilainya kosong (null)
- Kenapa nilainya kosong (null)? Karena kita belum mengasssign alamat memory ke variable pointer m1

Keyword “new”

- Bagaimana caranya agar m1 bisa dipakai?
- Pertama kita, harus assign alamat memori ke variable pointer m1.
- Sehingga kita harus memesan memory untuk menaruh objek mahasiswa tersebut menggunakan fungsi new() dan mengassign nilai kembalian dari fungsi new() (yang berisi alamat) ke variable pointer m1.

Contoh Kasus

```
1   Class Mahasiswa {
2       private String nama;
3       private int nim;
4       public void setNama(String nama) {
5           this.nama = nama;
6       }
7       public String getNama() {
8           return nama;
9       }
10      public void setNim(int nim) {
11          this.nim = nim;
12      }
13      public int getNim() {
14          return nim;
15      }
```

- Diberikan sebuah class Mahasiswa

Contoh Kasus

```
16      public static void main(String[] args) {
17          Mahasiswa m1;
18          m1 = new Mahasiswa();
19          m1.setNama("andi");
20          m1.setNim(1);
21          Mahasiswa m2;
22          m2 = m1;
23          m2.setNama("budi");
24          m2.setNim(99);
25
26          System.out.println("Nama : " +m1.getNama());
27          System.out.println("NIM : " +m1.getNim());
28      }
```

- Apa output dari program tersebut? (nilai dari m1.getNama() dan m1. getNim() di baris 26 dan 27)
- Program tersebut akan beroutput :
  Nama : budi
  NIM : 99
- Mengapa terjadi seperti itu?

Simulasi

```
Mahasiswa m1;
```

Membuat variable pointer of Mahasiswa.
Nilai dari m1 adalah NULL alias kosong

Simulasi

```
Mahasiswa m1;
m1 = new Mahasiswa();
```

Meminta alamat kepada memory manager untuk data/objek baru Mahasiswa (fungsi new).

Lalu menyimpan alamat tersebut di variable pointer m1

Misal kita diberi alamat 59,122

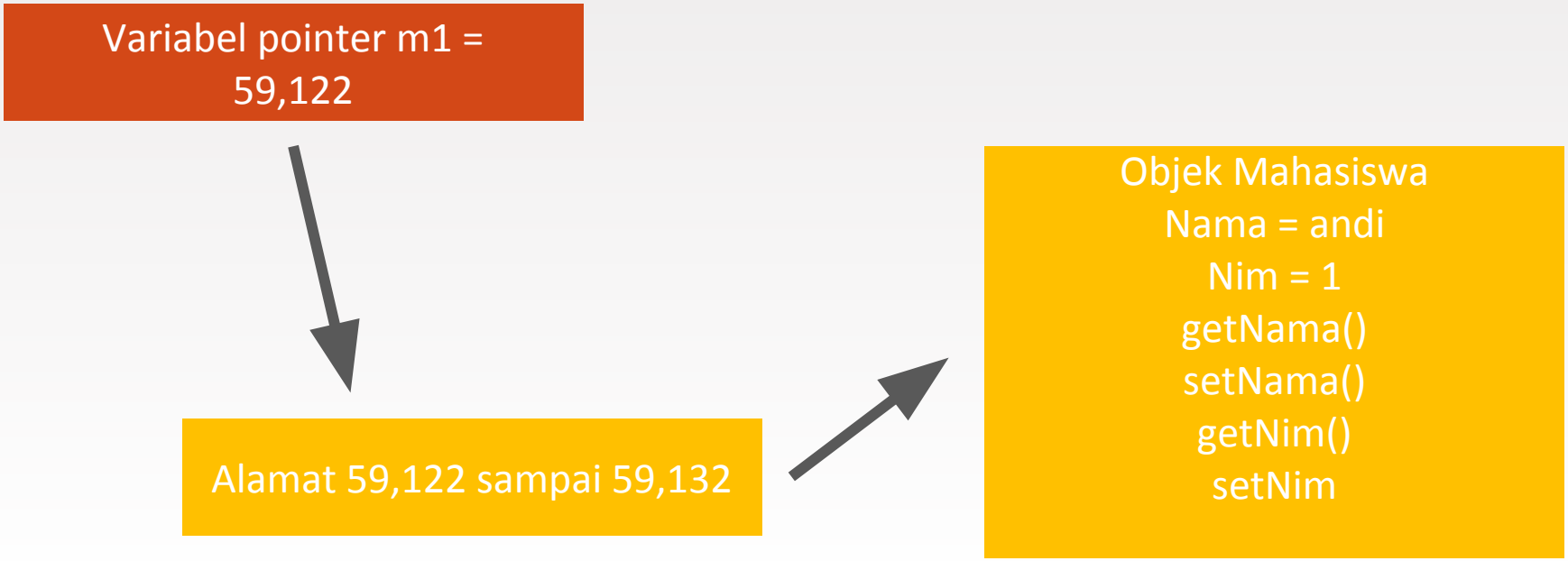
Anggap saja sebagai abstraksi jika ukuran objek Mahasiswa adalah 10 byte (ngasal)

Maka nilai variable m1 = 59,122

Simulasi

```
Mahasiswa m1;
m1 = new Mahasiswa();
m1.setNama("andi");
m1.setNim(1);
```

Mengeset property menggunakan setter dan getter

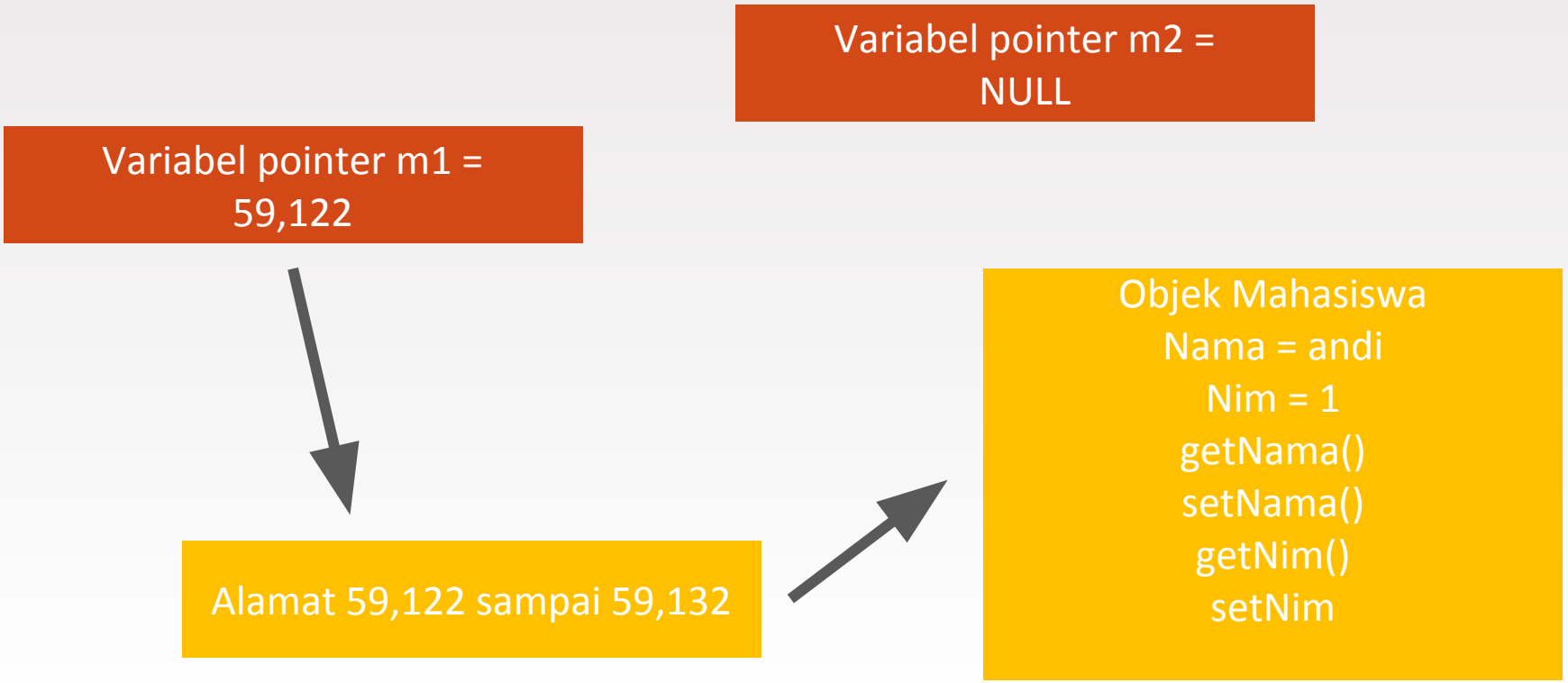


Simulasi

```
Mahasiswa m1;
m1 = new Mahasiswa();
m1.setNama("andi");
m1.setNim(1);
Mahasiswa m2;
```

Membuat variable pointer m2, bernilai null karena belum diisi

Sehingga m2 tidak menunjuk kemana-mana



Simulasi

```
Mahasiswa m1;
m1 = new Mahasiswa();
m1.setNama("andi");
m1.setNim(1);
Mahasiswa m2;
```

Apa yang terjadi ketika kita melakukan `m2.setNama("Asal")`

Akan terjadi error **null pointer exception** karena mengakeses method/property dari object yang bernilai null (seperti yang sudah dibahas tadi)

Variabel pointer m2 = NULL

Variabel pointer m1 = 59,122

Alamat 59,122 sampai 59,132

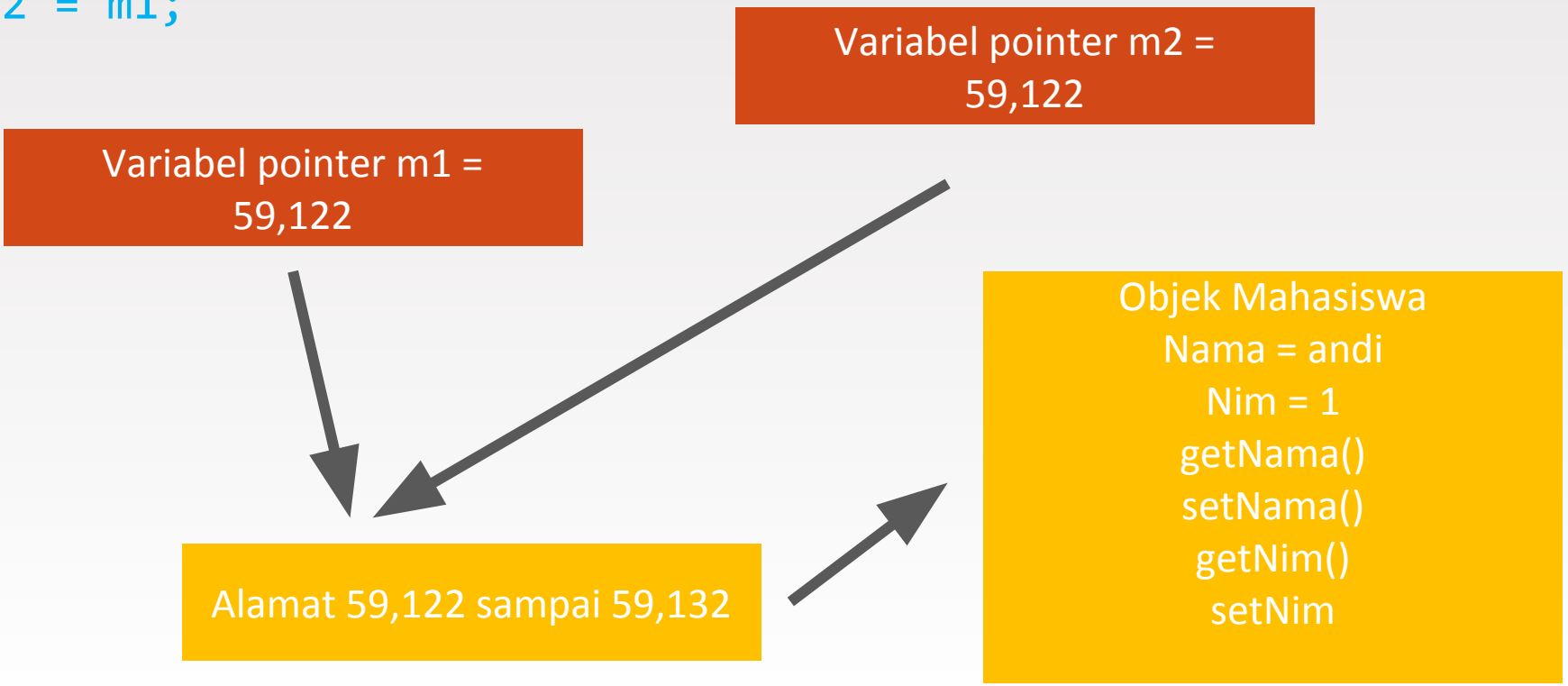
Objek Mahasiswa
Nama = andi
Nim = 1
getNama()
setNama()
getNim()
setNim

Simulasi

```
Mahasiswa m1;
m1 = new Mahasiswa();
m1.setNama("andi");
m1.setNim(1);
Mahasiswa m2;
m2 = m1;
```

Mengisi m2 dengan alamat 59,122.

Tidak ada objek baru yang dibuat ataupun tidak mengopi objek baru.



Simulasi

```
. . .
Mahasiswa m2;
m2 = m1;
m2.setNama("budi");
m2.setNim(99);
```

Merubah alamat objek mahasiswa yang ditunjuk oleh variable pointer m2, yaitu alamat 59,122