DASAR PEMROGRAMAN

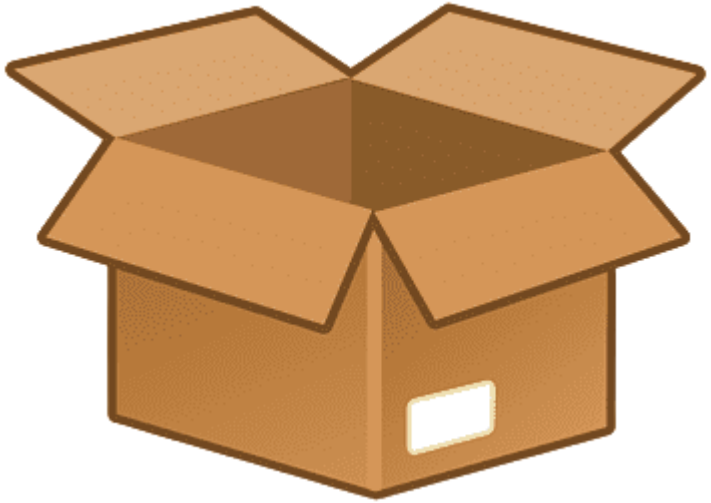
TIM AJAR

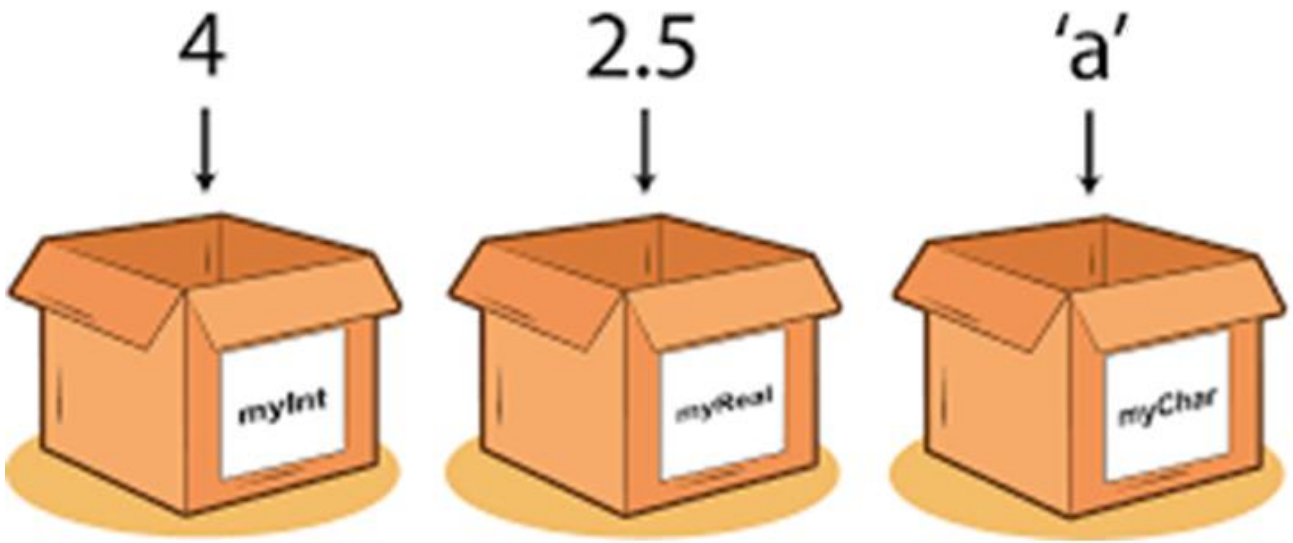
DASAR PEMROGRAMAN

2023/2024

Variable

- Variable merupakan sebuah tempat untuk menyimpan nilai sementara
- Nilai tersebut nantinya bisa diakses dan dimodifikasi
- Variable memiliki **data type** (tipe data) dan **nama**
- **Nama** merupakan identitas variable
- **Tipe data** menentukan jenis data yang dapat disimpan oleh variable tersebut




4
2.5
'a'
myInt
myReal
myChar

----------Declaration/Deklarasi----------

```
int jumlah;
double angka;
float a, b, c;
```

----------Assignment/Pemberian nilai--------

```
jumlah = 75;
jumlah = 50;
angka = 2.5;
```

Initialization → assignment untuk pertama kali

```
int a = 5;
int b = 10;
a = b;

System.out.println(a);
System.out.println(b);
```

Mencetak Variabel

```
int a = 7;
int b = 3;

System.out.println(“Jumlah = ” + a + b);
```

Jenis operator

1. Operator Aritmatika
2. Operator Assignment
3. Operator Increment dan Decrement
4. Operator Relasi
5. Operator Logika
6. Operator Bitwise

5. Operator Logika

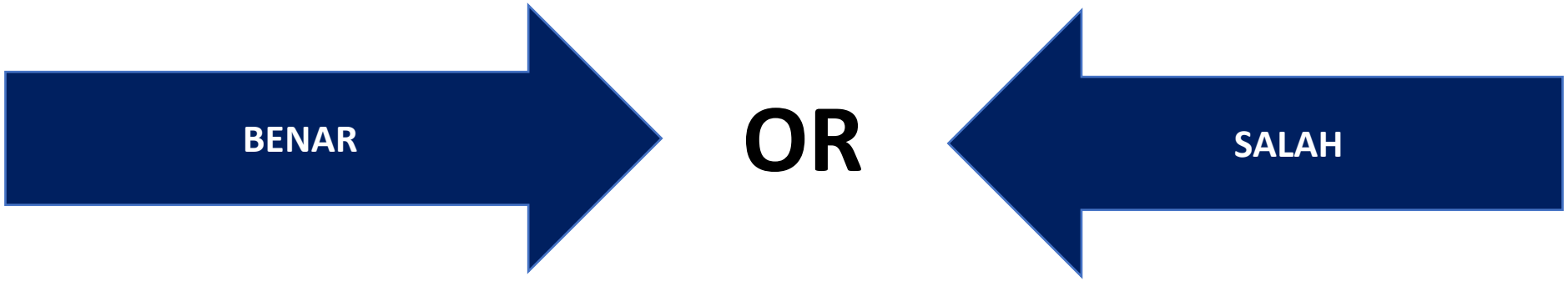
Operator ini digunakan untuk ekspresi logika yang menghasilkan nilai boolean.

Operator	Deskripsi	Contoh
&&	and	x=6 y=3 (x < 10 && y > 1) hasil true
||	or	x=6 y=3 (x==5 || y==5) hasil false
!	not	x=6 y=3 !(x==y) hasil true

PEMILIHAN

- Pemilihan(selection)adalah instruksi untuk yang dipakai untuk memilih satu kemungkinan dari beberapa kondisi

Kondisi : suatu pernyataan atau ekspresi (pernyataan logika)



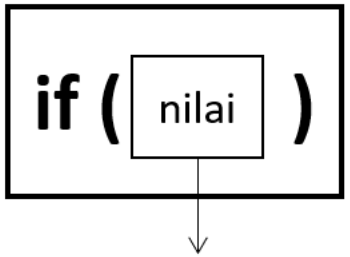
PEMILIHAN

- CONTOH :
  - **IF** your name starts with a ‘J’
  - **THEN** raise your right hand
  - **ELSE** sit down

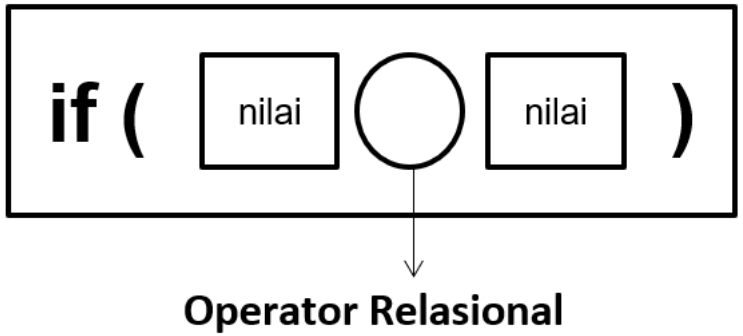
BENTUK SINTAKS PEMILIHAN

1. IF
2. IF…ELSE
3. IF…ELSE IF…ELSE…
4. SWITCH…CASE

Sintaks Pemilihan IF




```
if (isLapar) {
   Print("silakan makan")
}
```



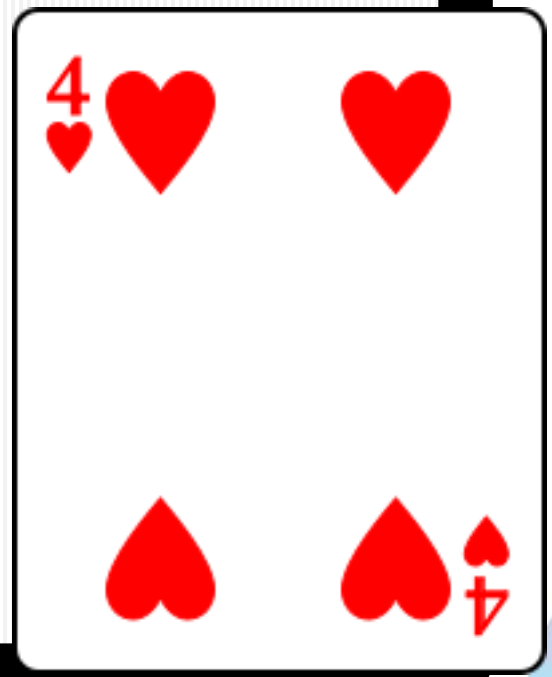

```
if (nilai < 70 ) {
    Print("harus remidi")
}
```

Struktur Pemilihan IF...ELSE

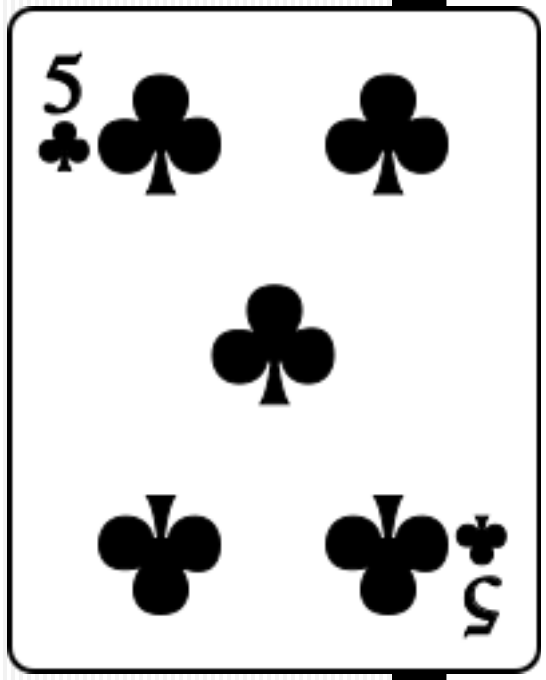
```
if (isLapar) {
   Print("silakan makan")
}
else{
    Print("tidak perlu makan")
}
```

```
if (nilai < 70 ) {
    Print("harus remidi")
}
else{
     Print("tidak perlu remidi")
}
```

```
if (warnaKartu == "merah" && nilaiKartu >= 5) {
    s.o.p("TEPUK TANGAN");
}
else{
    s.o.p("HENTAK KAKI");
}
```



```
if (warnaKartu == "merah" && nilaiKartu >= 5) {
    s.o.p("TEPUK TANGAN");
}
else if (warnaKartu == "hitam" && nilaiKartu >= 5) {
    s.o.p("TUNJUK TEMAN");
}
else{
    s.o.p("HENTAK KAKI");
}
```



```
System.out.print(s:"Masukkan jenis buah: ");
String buah = sc.nextLine();

switch (buah) {
    case "Jeruk":
        System.out.println(x:"Harga per kg Rp 15.000");
        break;
    case "Nanas":
        System.out.println(x:"Harga per kg Rp 10.000");
        break;
    case "Anggur":
    case "Apel":
        System.out.println(x:"Harga per kg Rp 30.000");
        break;
    default:
        System.out.println(x:"Maaf, harga belum terdaftar");
        break;
}
```

```
Masukkan jenis buah: Anggur
Harga per kg Rp 30.000
```

Pemilihan Bersarang

```
9   int bil1, bil2, bil3, max;
10
11  System.out.print("Bilangan 1: ");
12  bil1 = sc.nextInt();
13  System.out.print("Bilangan 2: ");
14  bil2 = sc.nextInt();
15  System.out.print("Bilangan 3: ");
16  bil3 = sc.nextInt();
17
18  if (bil1 > bil2) {
19      if (bil1 > bil3) {
20          max = bil1;
21      } else {
22          max = bil3;
23      }
24  } else {
25      if (bil2 > bil3) {
26          max = bil2;
27      } else {
28          max = bil3;
29      }
30  }
31
32  System.out.println("Max: " + max);
```

Definisi Perulangan

- Perintah perulangan atau iterasi (loop) adalah perintah untuk mengulang satu atau lebih statement sebanyak beberapa kali
- Loop statement digunakan agar kita tidak perlu menuliskan satu/sekumpulan statement berulang-ulang.
- Tujuannya adalah meringkas program dan menghindari kesalahan ketik
- Tipe perulangan:
  - Definite loop
  - Indefinite loop

Tipe Perulangan – Definite Loop

- Perulangan yang jumlah eksekusinya **telah diketahui sebelumnya**
- Biasanya ditandai dengan “ulangi sebanyak __ kali”
- Contoh:
  - Ulangi pernyataan ini sebanyak n kali
  - Ulangi pernyataan ini untuk setiap bilangan genap antara 8 dan 26

Tipe Perulangan – Indefinite Loop

- Perulangan yang jumlah eksekusinya **tidak dapat ditentukan sebelum dilakukan**
- Perulangan dieksekusi selama kondisi bernilai TRUE atau sampai kondisi bernilai FALSE
- Contoh:
  - Ulangi pernyataan ini selama bilangan n bukan bilangan prima
  - Ulangi pernyataan ini sampai pengguna memasukkan bilangan bulat yang valid

Jenis Perintah Perulangan

Dalam bahasa Java, ada 3 macam perintah perulangan yang umum digunakan yaitu:

- Perintah FOR
- Perintah WHILE
- Perintah DO-WHILE

Perulangan FOR

- FOR umumnya digunakan pada pengulangan yang jumlah perulangannya sudah pasti atau sudah diketahui sebelumnya
- Sintaks FOR

```
for (inisialisasi; kondisi; update) statement;
```

atau:

```
for (inisialisasi; kondisi; update){
        statement1;
        statement2;
        ……
}
```

```java
public static void main(String[] args) {
    System.out.println("Mata kuliah Dasar Pemrograman");
    System.out.println("Mata kuliah Dasar Pemrograman");
    System.out.println("Mata kuliah Dasar Pemrograman");
    System.out.println("Mata kuliah Dasar Pemrograman");
    System.out.println("Mata kuliah Dasar Pemrograman");
}
```

```java
for (int i = 0; i < 5; i++) {
    System.out.println("Mata kuliah Dasar Pemrograman");
}
```

```java
for (int i = 0; i < 5; i++) {
    System.out.println("Mata kuliah Dasar Pemrograman");
}
```

```java
for (int i = 25; i < 30; i++) {
    System.out.println("Mata kuliah Dasar Pemrograman");
}
```

```java
for (int i = 0; i < 5; i++) {
    System.out.println("Mata kuliah Dasar Pemrograman");
}
```

```java
for (int i = 5; i > 0; i--) {
    System.out.println("Mata kuliah Dasar Pemrograman");
}
```

```java
for (int i = 0; i < 5; i++) {
    System.out.println(i);
}
```

```
0
1
2
3
4
```

```
5
4
3
2
1
```

```
for (int i = 5; i > 0; i--) {
    System.out.println(i);
}
```

Perulangan WHILE

- WHILE cocok digunakan untuk perulangan yang jumlahnya tidak diketahui sebelumnya (indefinite loop)
- Sintaks WHILE

```
inisialisasi;

while (kondisi){
        statement1;
        statement2;
        …
        update;
}
```

Perulangan akan terus dijalankan selama `kondisi` bernilai TRUE

Perbandingan FOR dan WHILE

```
int x = 1;

while (x <= 10)
{
    ……
    ……
    x++;
}
```

setara

```
for(int x = 1;  x <= 10;  x++)
{
    ……
    ……
}
```

Perulangan DO-WHILE

- Sintaks DO-WHILE

```
inisialisasi;

do{
        statement1;
        statement2;
        …
        update;
} while (kondisi);
```

Eksekusi statement **minimal 1 kali.**
Selama `kondisi` bernilai TRUE,
maka perulangan akan terus dijalankan

Contoh Perulangan DO-WHILE

Kode Program

```
int x = 10;

do {
    System.out.println(x);
    x++;
} while (x < 5);
```

Output

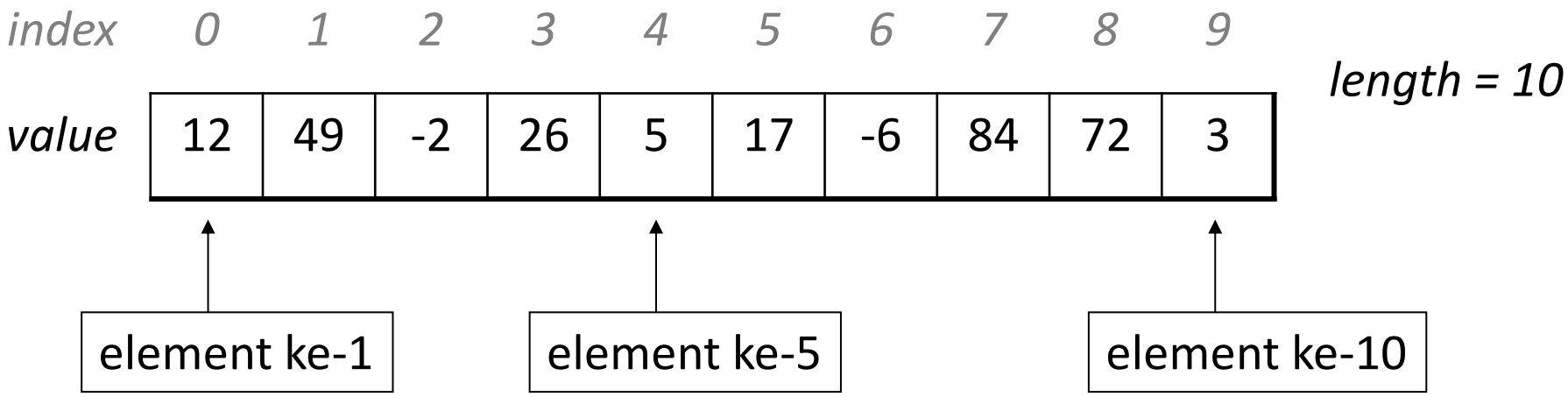
```
10
```

```
12          System.out.print("Jatah cuti: ");
13          jatahCuti = sc.nextInt();
14
15          do {
16              System.out.print("Apakah Anda ingin mengambil cuti (y/t)? ");
17              konfirmasi = sc.next();
18
19              if (konfirmasi.equalsIgnoreCase("y")) {
20                  System.out.print("Jumlah hari: ");
21                  jumlahHari = sc.nextInt();
22
23                  if (jumlahHari <= jatahCuti) {
24                      jatahCuti -= jumlahHari;
25                      System.out.println("Sisa jatah cuti: " + jatahCuti);
26                  } else {
27                      System.out.println("Sisa jatah cuti Anda tidak mencukupi");
28                      continue;
29                  }
30              }
31              else{
32                  break;
33              }
34          } while (jatahCuti > 0 );
```

Array

- Array merupakan suatu struktur data yang berisi sekumpulan nilai (elemen) dengan tipe data yang sama.
- Masing-masing elemen array bisa diakses dengan menggunakan indeks yang unik

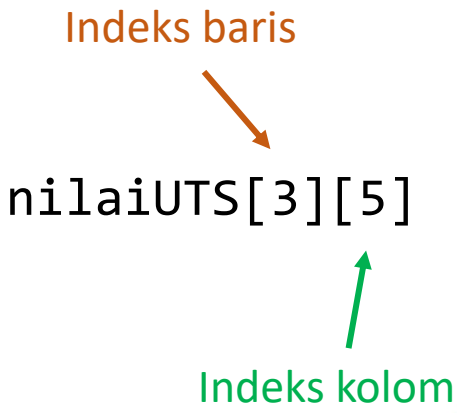
Visualisasi Array



Array 2 Dimensi

- Array 2 dimensi dapat digunakan untuk menyimpan data yang terdiri **beberapa baris** dan **beberapa kolom** ke dalam sebuah variabel array
- Sama halnya dengan array satu dimensi, array 2 dimensi juga mempunyai nomor indeks, namun nomor indeks terdiri dari 2 angka

Mahasiswa (Baris) \ Mata Kuliah (Kolom)	0	1	2	3	4	5	6	7
0	79	87	94	88	67	81	75	92
1	63	83	58	80	86	69	98	87
2	84	88	60	82	80	74	84	75
3	70	91	65	94	80	91	85	60
4	93	84	77	97	76	82	73	91



Default Value

- Seperti halnya pada array 1 dimensi, instansiasi array 2 dimensi (dengan keyword new) memberikan nilai default untuk setiap elemennya
  - String → null
  - int, double → 0
  - boolean → false

```
int[][] x = new int[3][5];
```

	0	1	2	3	4
0	0	0	0	0	0
1	0	0	0	0	0
2	0	0	0	0	0

```
boolean[][] y = new boolean[2][3];
```

	0	1	2
0	False	False	False
1	False	False	False

```
String[][] z = new String[3][2];
```

	0	1
0	*null*	*null*
1	*null*	*null*
2	*null*	*null*

Ukuran Array 2 Dimensi

- Setiap array, baik array 1 dimensi atau 2 dimensi, mempunyai ukuran
- Ukuran array dapat diketahui dengan atribut length
- Contoh:

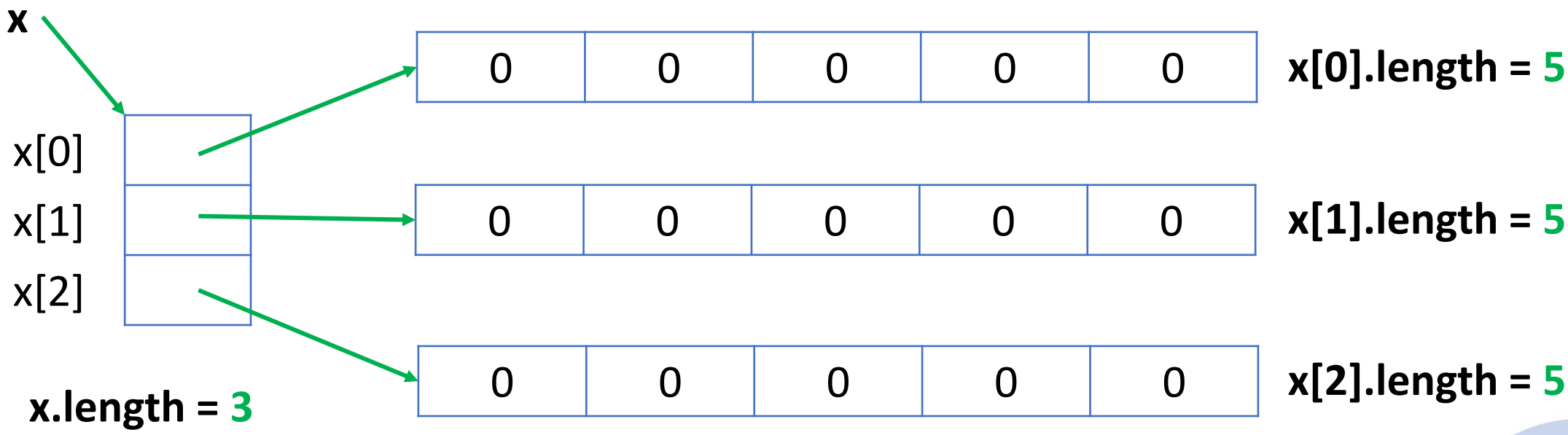
```
int[][] x = new int[3][5];
```

x.length menghasilkan jumlah barisnya (dimensi pertama) yaitu 3

x[0].length menghasilkan jumlah kolomnya (dimensi kedua) yaitu 5

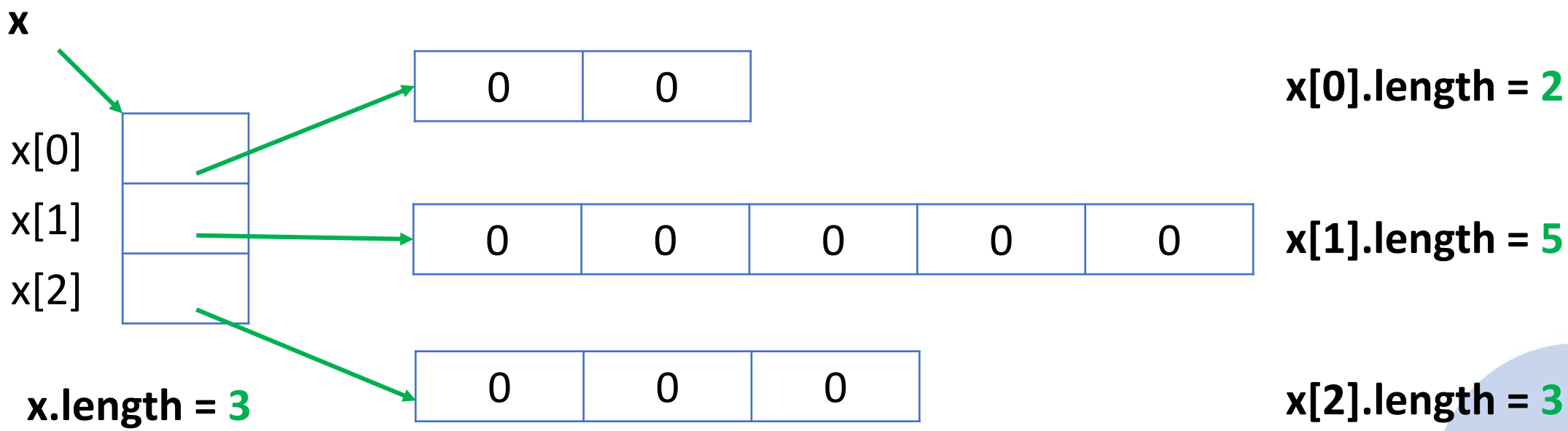
Ukuran Array 2 Dimensi (2)

```
int[][] x = new int[3][5];
```



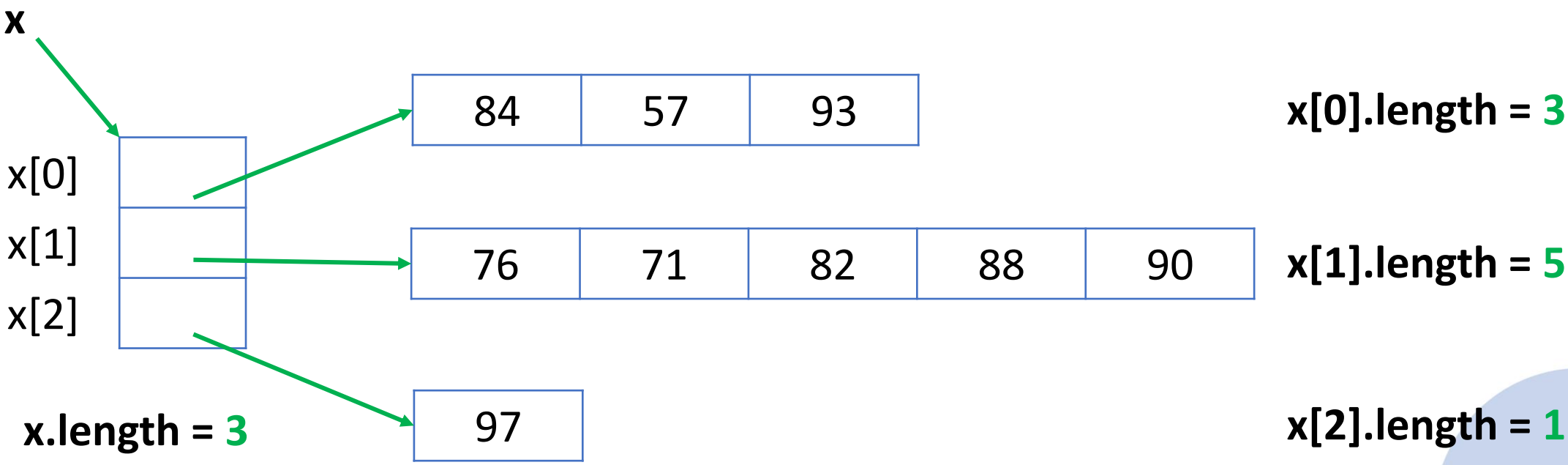
Ukuran Array 2 Dimensi (3)

```
int[][] x = new int[3][];
x[0] = new int[2];
x[1] = new int[5];
x[2] = new int[3];
```



Ukuran Array 2 Dimensi (4)

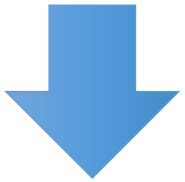
```
int[][] x = {
    {84, 57, 93},
    {76, 71, 82, 88, 90},
    {97}
};
```



```
int[][] nilai = {
      {84, 57, 93},
      {76, 71, 82, 88, 90},
      {97}
};
```

	0	1	2	3	4
0	84	57	93		
1	76	71	82	88	90
2	97				

```
nilai[2][2] = 1
System.out.print(nilai[2][3]);
```

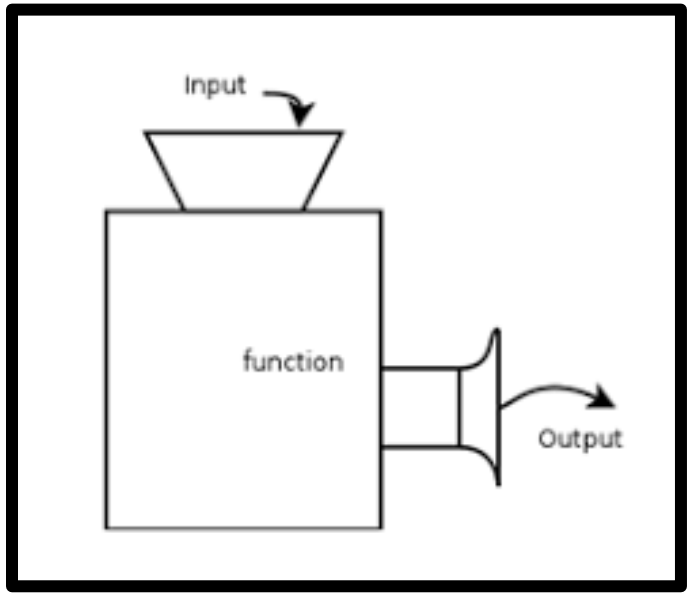


ArrayIndexOutOfBoundsException

array diakses dengan illegal index

DEFINISI FUNGSI

- Fungsi (function) adalah sejumlah instruksi yang dikelompokkan menjadi satu, berdiri sendiri, yang berfungsi untuk menyelesaikan suatu pekerjaan tertentu.
- Input dan output tidak wajib ada
- Jika menggunakan fungsi maka program dapat disusun secara lebih terstruktur (lebih modular) dan lebih efektif.



MANFAAT FUNGSI

- **Modularity** → memecah program menjadi modul-modul dengan tugas yang independen sehingga lebih mudah dikelola
- **Effectiveness** → block statement tidak perlu ditulis berulang kali

DEKLARASI FUNGSI

```
static <returnType> <namaFungsi>() {
      // statement
      // statement
}
```

Keterangan:

- **Static**: fungsi dideklarasikan sebagai static agar tidak perlu dilakukan instansiasi objek terlebih dahulu
- **Return Type**: tipe data dari return value (nilai yang dikembalikan/*output* setelah fungsi dieksekusi)

1

a. Fungsi tanpa parameter
b. Fungsi dengan parameter

Parameter

- Parameter adalah variable khusus yang digunakan untuk menampung nilai input pada suatu fungsi
- Fungsi boleh tidak memiliki parameter
- Parameter ada jika diperlukan data yang asalnya dari luar fungsi
- Jumlah parameter sesuai dengan kebutuhan dan tidak ada jumlah maksimalnya
- Pada saat deklarasi fungsi, penulisan parameter adalah dengan cara:

```
<tipe_data_parameter> <nama _parameter>
```

1.a. Fungsi tanpa Parameter

- Jika fungsi tidak memerlukan input, *parenthesis* (...) setelah nama fungsi kosong

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

static void printWelcomeMessage(){
    System.out.println("Welcome, Guest!");
}
```

```
Welcome, Guest!
```

1.b. Fungsi dengan Parameter (1)

- Setiap parameter terdiri dari tipe data dan nama parameter (misal: int a, float b), sama persis seperti deklarasi variabel.
- Parameter ditulis di dalam *parenthesis* (...) setelah nama fungsi.
- Bila terdapat lebih dari satu parameter, maka dipisah dengan tanda koma dan masing-masing parameter harus dideskripsikan tipe datanya.
- Jika fungsi memiliki parameter, maka ketika dipanggil harus menyertakan argumen
- Argumen adalah nilai yang dilempar ketika fungsi dipanggil

```
static <returnType> <namaFungsi>(<tipeData1> <namaParameter1>, <tipeData2> <namaParameter2>)
{
        // statement
        // statement
}
```

1.b. Fungsi dengan Parameter (2)

```
public static void main(String[] args) {
    printWelcomeMessage("Ani");
    printWelcomeMessage("Budi");
    printWelcomeMessage("Cica");
}

static void printWelcomeMessage(String name){
    System.out.println("Welcome, " + name + "!");
}
```

Argumen

Parameter

```
Welcome, Ani!
Welcome, Budi!
Welcome, Cica!
```

2

a. Fungsi dengan nilai kembalian (return value)
b. Fungsi tanpa nilai kembalian (no return value)

2.a. Fungsi tanpa Return Value

- Fungsi yang tidak memiliki return value, maka return type-nya dideklarasikan sebagai **void**

```
static void <namaFungsi> ()
{
        //statement
        //statement
}
```

2.b. Fungsi dengan Return Value

- Sebuah fungsi yang dapat mengembalikan nilai *output* sehingga bisa dipakai pada proses berikutnya
- Pengembalian nilai pada fungsi menggunakan *keyword* **return**
- Fungsi yang memiliki **return type fungsi selain void** berarti harus memiliki keyword **return**
- **Nilai yang di-return harus sesuai dengan return type**. Misalnya return type int, maka nilai yang di-return harus bertipe int pula.

2.b. Fungsi dengan Return Value

```
static <returnType> <namaFungsi>(){
      // statement
      return variabelOutput;
}
```

2.b. Fungsi dengan Return Value

```
public static void main(String[] args) {
    int luasPersegi = getLuasPersegi(5);
    System.out.println("Luas persegi dengan sisi 5 adalah " + luasPersegi);
}

static int getLuasPersegi(int sisi) {
    int luas = sisi * sisi;
    return luas;
}
```

```
Luas persegi dengan sisi 5 adalah 25
```