
# Back End Development 1

## Sesi 4

# Array & Structure Data Tree

# Introduction

Array adalah sebuah variabel yang bisa menyimpan banyak data dalam 1 variabel dan menggunakan indeks untuk mengakses data.

Array yang ada di Java adalah sebuah objek yang berisikan elemen-elemen dari tipe data yang mirip.

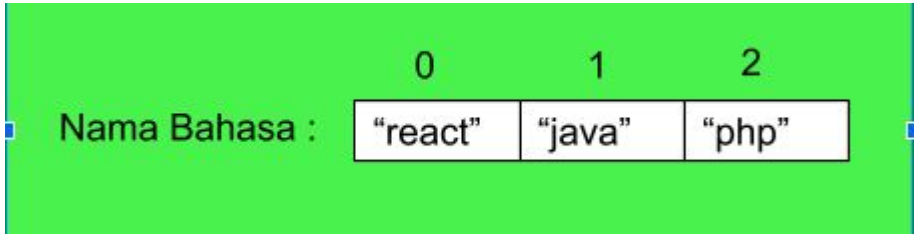
Tidak hanya itu, elemen yang ada di dalam array ini juga disimpan di lokasi memori yang berdekatan.

Array ini adalah struktur data tempat untuk menyimpan elemen yang serupa. Anda hanya dapat menyimpan satu set elemen tetap dalam setiap array Java.

Array pada Java berbasis indeks, di mana elemen pertama dari array ini disimpan di dalam indeks 0, elemen ke-2 disimpan di indeks pertama dan begitulah seterusnya.

Tidak seperti bahasa C / C++, Anda bisa mendapatkan panjang array dengan menggunakan panjang member.

Sementara dalam bahasa C / C++, Anda perlu menggunakan operator sizeof.




Dalam Java, array adalah sebuah objek yang menghasilkan kelas secara dinamis. Anda dapat menyimpan nilai atau objek primitif dalam sebuah array di Java. Seperti bahasa C/C++, Anda juga dapat membuat array dimentional atau multidimentional di Java. Selain itu, Java menyediakan fitur array anonym yang tidak tersedia di C/C++.

# Deklarasi Array

Perlu diingat bahwa Array pada Java adalah sebuah objek juga. Array perlu dideklrasikan dan kemudian dibuat.

Mendeklarasikan variabel yang akan menampung array bilangan bulat, kita menggunakan sintaks sebagai berikut:

```
int[] contoh;
```

Jika Anda perhatikan sintaks di atas, tidak terdapat ukuran di dalamnya. Hal ini dikarenakan kita belum membuat array.

Kemudian pada sintaks berikut ini kami akan membuat sebuah array baru dengan ukuran 4.

```
contoh = new int[4];
```

Anda dapat mengecek ukurannya dengan mencetak panjang array.

```
System.out.println(contoh.length);
```

Kita dapat mengakses array dan melakukan setting nilainya.

```
contoh[0] = 5;
contoh[1] = 10;
contoh[2] = contoh[1] + 10;
```

Seperti yang sudah dijelaskan sebelumnya, array Java memiliki dasar 0, yang berarti elemen pertama dari sebuah array diakses dalam index 0.

Sebagai contoh, sebuah aray dengan ukuran 4 hanya akan meningkat ke index 3 karena ia berbasis 0.

```
int[] contoh = new int[4];
//mengakses dan memberi nilai elemen terakhir
arr[3] = 9;
```

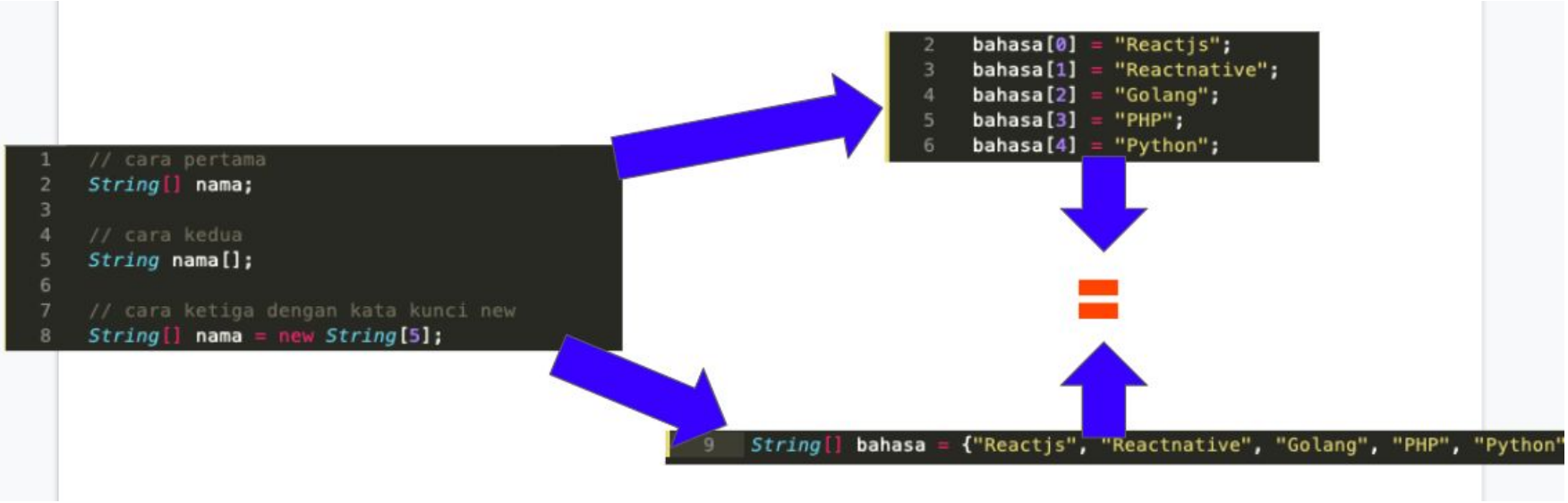
Anda juga dapat membuat sebuah array dengan nilai yang sama dalam line.

```
int[] contoh = {5, 15, 19, 9};
```

# Cara Kerja Array

Bagaimana Array Bekerja ?

Perhatikan :
- Kita menggunakan [ ] untuk membuat array
- Kurung siku diletakkan setelah tipe data atau nama array
- Angka 5 dalam kurung artinya batas atau ukuran array-nya
- Array yang kosong siap diisi dengan data. Pastikan mengisinya dengan data yang sesuai dengan tipe datanya.


// cara pertama
String[] nama;
// cara kedua
String nama[];
// cara ketiga dengan kata kunci new
String[] nama = new String[5];
bahasa[0] = "Reactjs";
bahasa[1] = "Reactnative";
bahasa[2] = "Golang";
bahasa[3] = "PHP";
bahasa[4] = "Python";
String[] bahasa = {"Reactjs", "Reactnative", "Golang", "PHP", "Python"

# Array Handling

Array memiliki indeks, sehingga mempermudah kita untuk mengambil data nya. Contohnya.

```
18    // mengambil data array
19    System.out.println(bahasa[2]);
```

Apa hasilnya ?

# Array Looping

Disini kita menggunakan atribut length untuk mengambil panjang arraynya. Jadi, perulangan akan dilakukan sebanyak isi array-nya.

```
22  package array2;
23
24  public class ContohArray {
25      public static void main(string[] args) {
26          String[] bahasa ={"Reactjs","Reactnative","Golang","PHP"};
27
28          for(int i=0 ; i < bahasa.;length, i++){
29              system.out.printIn("indeks ke-"+1+":" + bahasa[i]);
30          }
31      }
32  }
```

# Array Looping - FOR EACH

Seperti yang sudah kita pelajari pada perulangan for-each, yang digunakan untuk mengambil data.

```
34  import java.util.Scanner;
35
36  public class itprofesi {
37      public static void main(String[] args) {
38
39          // membuat array it profesi
40          String[] itprofesi = new String[5];
41
42          // membuat scanner
43          Scanner scan = new Scanner(System.in);
44
45          // mengisi data ke array
46          for( int i = 0; i < itprofesi.length; i++ ){
47              System.out.print("Buah ke-" + i + ": ");
48              itprofesi[i] = scan.nextLine();
49          }
50
51          System.out.println("----------------------------");
52
53          // menampilkan semua isi array
54          for( String b : itprofesi ){
55              System.out.println(b);
56          }
57
58      }
59  }
```

# Array MultiDimensi

Array yang memiliki lebih dari 1 dimensi atau bisa disebut array dalam array.

```
69  String[][] framework = {
70      {"Java","kotlin"},
71      {"React","Reactjs"},
72      {"Php","Laravel"}
```

**Array 2 Dimensi**

| | 0 | 1 |
|---|---|---|
| 0 | Java | Kotlin |
| 1 | React | Reactjs |
| 2 | Php | Laravel |

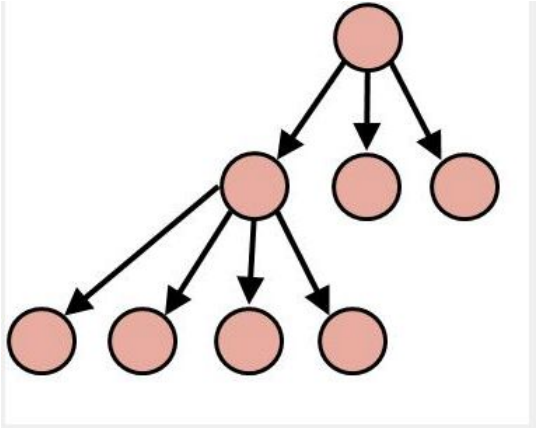
Indeks ke-0 pada array framework berisi array {“Java”, “Kotlin”}

**How to Access Array MultiDimensi**

```
61  Package arraymulti;
62
63  public class array2D {
64      public static void main(String[] args){
65          String[][] framework = {{"Java","kotlin"},{"React","Reactnative"},{"Php","Laravel"}};
66
67          for (int x=0; x<framework.length; x++){
68              System.out.println("bahasa:" + framework[x][0]);
69              System.out.println("framework:" + framework[x][1]);
70              System.out.println("-------------------------");
71          }
72      }
73  }
```

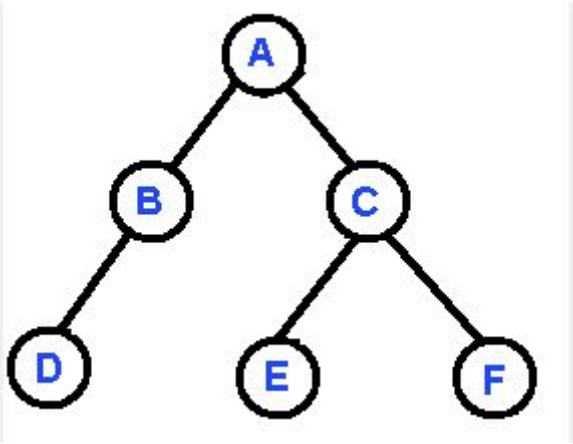
# Data Structure - TREE

# Introduction TREE



Tree (pohon) adalah salah satu bentuk struktur data yang menggambarkan hubungan hierarki antar elemen-elemennya (seperti relasi one to many).

Sebuah node dalam tree biasanya bisa memiliki beberapa node lagi sebagai percabangan atas dirinya.

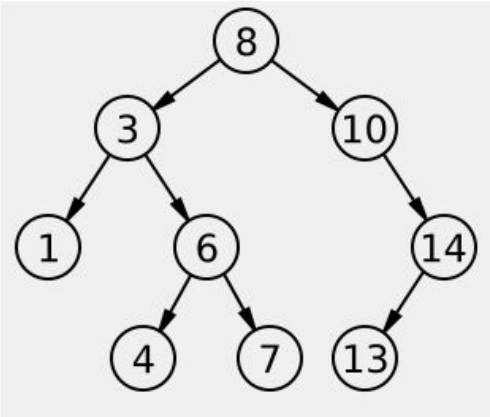



Lalu, ada lagi yang namanya Binary Tree. Apa bedanya? Sebenarnya sama sama konsepnya dengan Tree.

Hanya saja, kita akan mengambil sifat bilangan biner yang selalu bernilai 1 atau 0 (2 pilihan).

Berarti, binary tree adalah tree yang hanya dapat mempunyai maksimal 2 percabangan saja.

# Binary Search Tree




Struktur data yang mengadopsi konsep Binary Tree

Terdapat aturan bahwa setiap child node sebelah kiri selalu lebih kecil nilainya dari pada root node.

Begitu pula sebaliknya, setiap child node sebelah kanan selalu lebih besar nilainya daripada root node.

Kenapa harus membedakan kiri dan kanan sesuai besaran nilainya? **Tujuannya untuk memberikan efisiensi terhadap proses searching.**

“**Kalau struktur data tree sudah tersusun rapi sesuai aturan mainnya, proses search akan lebih cepat.**”

Aturan main Binary Search Tree :

- Setiap child node sebelah kiri harus lebih kecil nilainya daripada root nodenya.
- Setiap child node sebelah kanan harus lebih besar nilainya daripada root nodenya.

Lalu, ada 3 jenis cara untuk melakukan penelusuran data (traversal) pada BST :

- PreOrder : Print data, telusur ke kiri, telusur ke kanan
- InOrder : Telusur ke kiri, print data, telusur ke kanan
- Post Order : Telusur ke kiri, telusur ke kanan, print data

```
1.input
2.view
3.exit
 : 1
Masukkan Angka : 5

1.input
2.view
3.exit
 : 1
Masukkan Angka : 3
3 di kiri  5

1.input
2.view
3.exit
 : 1
Masukkan Angka : 9
9 di kanan 5

1.input
2.view
3.exit
 : 1
Masukkan Angka : 2
2 di kiri  3

1.input
2.view
3.exit
 : 1
Masukkan Angka : 4
4 di kanan 3

1.input
2.view
3.exit
 : 1
Masukkan Angka : 7
7 di kiri  9

1.input
2.view
3.exit
 : 1
Masukkan Angka : 11
11 di kanan 9
```

```
1.input
2.view
3.exit
 : 2
Pre Order   : 5 3 2 4 9 7 11
In Order    : 2 3 4 5 7 9 11
Post Order  : 2 4 3 7 11 9 5

1.input
2.view
3.exit
 : 3
Keluar
```

```
import java.util.Scanner;
public class tree{
    static Scanner in=new Scanner(System.in);
    public void insert(node a, int b){
        if(b<a.value){
            if(a.left!=null) insert(a.left,b);
            else{
                a.left=new node();
                a.left.input(b);
                System.out.println(b+" di kiri  "+a.value);
            }
        }
        else if(b>a.value){
            if(a.right!=null) insert(a.right,b);
            else{
                a.right=new node();
                a.right.input(b);
                System.out.println(b+" di kanan "+a.value);
            }
        }
    }
    public void view(node a){
        System.out.print("Pre Order  : ");
        preOrder(a);
        System.out.println(" ");
    }
public void preOrder(node a){
        if(a!=null){
            System.out.print(a.value+" ");
            preOrder(a.left);
            preOrder(a.right);
        }
    }
}
```

```
class node{
    node left,right;
    int value;
    public void input(int a){
        value=a;
    }
    public static void main(String[] args){
        tree tr=new tree();
        node root=new node();
        int menu=1;
        int r=1;
        int a;
        while(menu!=3){
            System.out.print("1.input\n2.view\n3.exit\n : ");
            menu=tr.in.nextInt();
            if(menu==1){
                System.out.print("Masukkan Angka : ");
                a=tr.in.nextInt();
                if(r==1){
                    root.input(a);
                    r--;
                }
                else tr.insert(root,a);
            }
            else if(menu==2) tr.view(root);
            else if(menu==3) System.out.println("Keluar");
            else System.out.println("Salah");
            System.out.println(" ");
        }
    }
}
```