

# Pertemuan 6

# Method, Class dan

# Object pada Java

Elia Dania Serepina  
PJ AP2A

# Method pada Java

**Metode (Method)** adalah sekumpulan statement program yang disatukan menjadi sebuah subprogram atau fungsi, diawali dengan tanda "{" diakhiri dengan tanda "}".

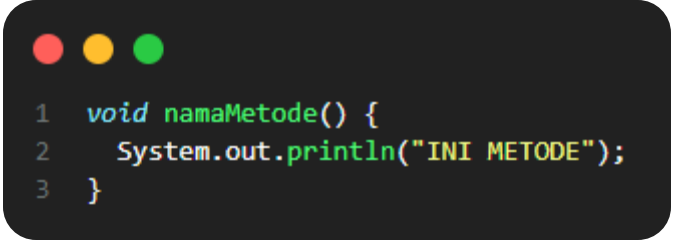
Ada 2 macam metode dan 1 metode pengendali, yaitu:

- **Metode kelas** : Metode ini dapat dieksekusi walaupun tidak terdapat objek dalam kelas tersebut. Seperti variabel kelas, metode kelas juga dideklarasikan menggunakan keyword static.
- **Metode objek** : Metode ini hanya dapat dieksekusi sehubungan dengan objek tertentu.
- **Metode main()** : Metode ini digunakan pada saat aplikasi Java dimulai, menggunakan keyword static. Sebelum aplikasi mulai dieksekusi, diperlukan metode walaupun tanpa objek.

Metode adalah suatu blok dari program yang berisi kode dengan nama dan properti yang dapat digunakan kembali. Metode dapat mempunyai nilai balik atau tidak

# Metode Tidak Membalikkan Nilai

Jika diberi awalan dengan kata void maka metode tersebut tidak memberi nilai balik.

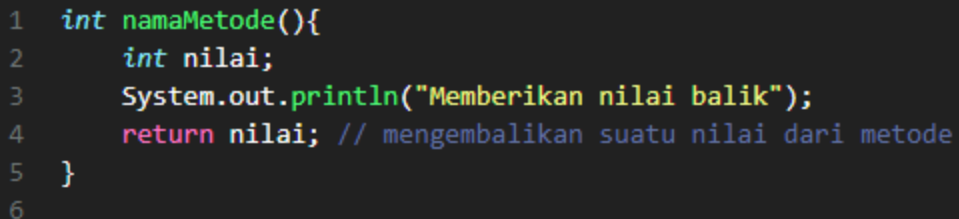


```
1 void namaMetode() {  
2     System.out.println("INI METODE");  
3 }
```

# Metode

## Membalikkan Nilai

jika metode diberi awalan sebuah tipe data maka metode tersebut akan memberi nilai balik data yang bertipedata sama dengan metode tersebut..

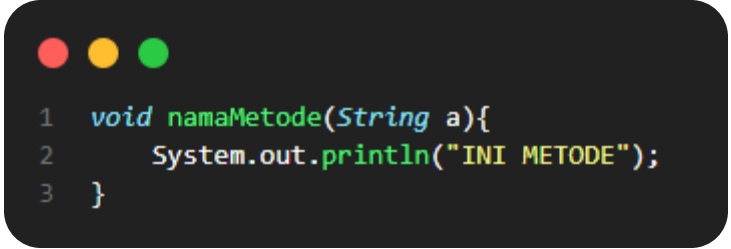


```
1  int namaMetode(){  
2      int nilai;  
3      System.out.println("Memberikan nilai balik");  
4      return nilai; // mengembalikan suatu nilai dari metode  
5  }  
6
```

# Metode

## Dengan Parameter

jika metode diberi awalan sebuah tipe data maka metode tersebut akan memberi nilai balik data yang bertipe data sama dengan metode tersebut..



```
1 void namaMetode(String a){  
2     System.out.println("INI METODE");  
3 }
```

# Karakteristik Method

Berikut adalah karakteristik dari method :

1. Dapat mengembalikan satu nilai atau tidak sama sekali
2. Dapat diterima beberapa parameter yang dibutuhkan atau tidak ada parameter sama sekali.
3. Setelah method telah selesai dieksekusi, dia akan kembali pada method yang memanggilnya.

# Memanggil Instance

Untuk memanggil sebuah instance method, kita dapat menuliskan :

***nameOfObject.nameOfMethod(parameters);***

Mari kita mengambil dua contoh method yang ditemukan dalam class String.

Deklarasi method	Definisi
<code>public char charAt(int index)</code>	Mengambil karakter pada indeks tertentu.
<code>public boolean equalsIgnoreCase (String anotherString)</code>	Membandingkan antar String, tidak case sensitive.

```
1 String str1 = "Hello";
2 char x = str1.charAt(0);
3 String str2 = "hello";
4 boolean result = str1.equalsIgnoreCase(str2);
```

# Pemberian Variabel pada Method

Ada dua tipe data variabel passing pada method, yang pertama adalah ***pass-by-value*** dan yang kedua adalah ***pass-by-reference***.

```
1 public class TestPassByValue {
2     public static void main( String[] args ){
3         int i = 10;
4         //mencetak nilai i
5         System.out.println( i );
6         //memanggil method test
7         //passing i pada method test test( i );
8         //Mencetak nilai i
9         System.out.println( i );
10    }
11
12    public static void test(int j){ //merubah nilai parameter j
13        j = 33;
14    }
15 }
```

```
1 class TestPassByReference {
2     public static void main( String[] args ){
3         //membuat array integer
4         int []ages = {10, 11, 12};
5
6         //mencetak nilai array
7         for( int i=0; i<ages.length; i++){
8             System.out.println( ages[i] );
9         }
10    }
11    //menambahkan method test
12    test( ages );
13
14    for( int i=0; i<ages.length; i++){
15        System.out.println( ages[i] );
16    }
17
18    public static void test( int[] arr ){ //merubah nilai array
19        for( int i=0; i<arr.length; i++){
20            arr[i] = i + 50;
21        }
22    }
23 }
```

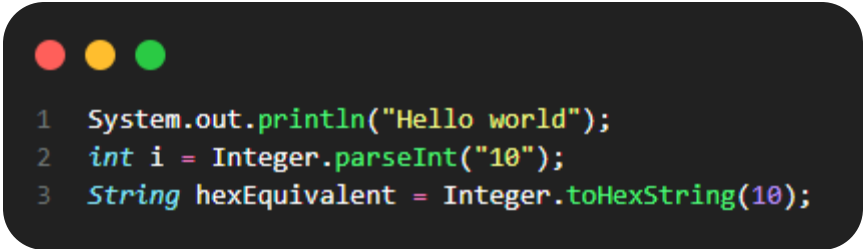


# Memanggil Method Static

**Method Static** adalah method yang dapat dipakai tanpa harus menginisialisasi suatu class (maksudnya tanpa menggunakan variabel terlebih dahulu). Method static hanya dimiliki oleh class dan tidak dapat digunakan oleh instance (atau objek) dari suatu class. Method static dibedakan dari method yang dapat instance di dalam suatu class oleh kata kunci static.

Untuk memanggil method static, ketik :

***Classname.staticMethodName(params);***



```
1 System.out.println("Hello world");  
2 int i = Integer.parseInt("10");  
3 String hexEquivalent = Integer.toHexString(10);
```

# Konstruktor

**Konstruktor** adalah suatu metode yang dapat digunakan untuk memberi nilai awal pada saat objek diciptakan. Konstruktor akan dipanggil secara otomatis begitu objek diciptakan. Konstruktor memiliki ciri :

- a. Namanya sama dengan nama kelas
- b. Tidak mengembalikan nilai ( dan juga tidak boleh ada kata void didepannya)

Jika constructor tidak didefinisikan, Java memberikan constructor dengan nama `constructor_default`. Constructor default tidak melakukan apa-apa, namun semua variabel yang diinisialisasi dianggap sebagai berikut:

- Variabel numerik diset ke 0
- String diset ke null
- Variabel boolean di set ke false

Constructor tidak memiliki tipe hasil, walaupun constructor bisa `public`, `private`, atau `protected`. Sebagian constructor bersifat `public`.

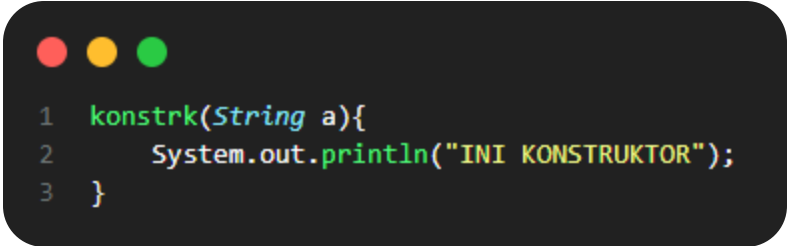
# Konstruktor



```
1  class Coba {  
2      Coba() { //Ini konstruktor  
3          System.out.println("Ini Konstruktor");  
4      }  
5  
6      public static void main(String[] args){  
7          Coba obj = new Coba();  
8      }  
9  }
```

# Konstruktor

Jika konstruktor dipanggil dari kelas turunan, maka caranya adalah dengan menuliskan kata **super();** pada class turunan. Konstruktor juga ada yang diberi parameter.

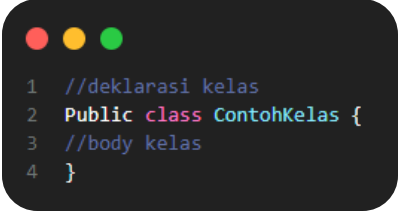


```
1  konstrk(String a){  
2      System.out.println("INI KONSTRUKTOR");  
3  }
```

# Kelas

**Kelas** berfungsi sebagai blueprint atau cetak biru yang mendefinisikan atribut dan perilaku objek. Dalam kelas, kita dapat menentukan karakteristik objek dengan mendefinisikan variabel (atribut) yang merepresentasikan keadaan objek serta fungsi (metode) yang menggambarkan perilaku objek tersebut.

Definisi kelas terdiri atas dua komponen, yaitu deklarasi kelas dan body kelas. Deklarasi kelas adalah baris pertama di suatu kelas, dan minimal mendeklarasikan nama kelas. Sementara itu, body dideklarasikan setelah nama kelas dan berada diantara kurung kurawal.



```
1 //deklarasi kelas
2 Public class ContohKelas {
3 //body kelas
4 }
```

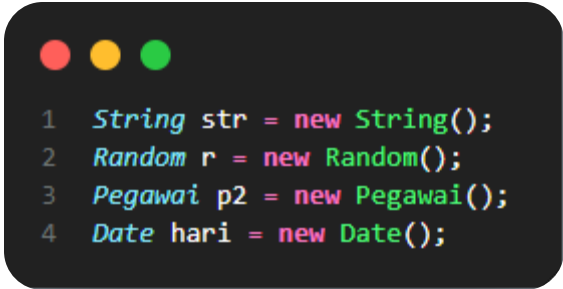
Pada Java, nama kelas sekaligus merepresentasikan nama file kode program dan sifatnya case sensitive.

# Objek

**Objek** adalah instansi konkret dari suatu kelas. Setiap objek memiliki atribut dan perilaku yang telah ditentukan oleh kelas yang menjadi dasarnya. Objek dianggap sebagai representasi nyata dari konsep atau entitas di dunia nyata yang didefinisikan oleh kelas.

Pada pemrograman java, cara untuk menciptakan sebuah objek dari suatu class adalah dengan cara sebagai berikut :

*<nama class> <nama objek> = new <nama konstruktor>*



```
1 String str = new String();  
2 Random r = new Random();  
3 Pegawai p2 = new Pegawai();  
4 Date hari = new Date();
```

# Instansiasi Class

Untuk membuat sebuah objek atau sebuah instance pada sebuah class. Kita menggunakan operator ***new***.

Sebagai contoh, jika anda ingin membuat instance dari class string, kita menggunakan kode berikut:

```
String str2 = new String("Hello world!");
```

Ini juga sama dengan,

```
String str2 = "Hello";
```

# Operator InstanceOf

**InstanceOf** memiliki dua operand: obyek pada sebelah kiri dan nama class pada sebelah kanan. Pernyataan ini mengembalikan nilai true atau false tergantung pada benar/salah obyek adalah sebuah instance dari penamaan class atau beberapa subclass milik class tersebut.



```
1  boolean ex1 = "Texas" instanceof String; // true
2  Object pt = new Point(10, 10);
3  boolean ex2 = pt instanceof String; // false
```



# Thanks!

