

# Pemrograman Berorientasi Objek

Metode, Deklarasi Paket dan Import Paket dalam Java



Dosen Pengampu :  
Noverly Lysbetti Marpaung, S.T., M.Sc.

**Kelompok 8 :**

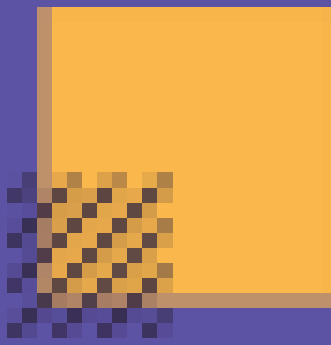
**Kartika Natalia Tumangger (2207112579)**

**Program Studi Teknik Informatika**

**Universitas Riau**

**Pekanbaru**

**2023**





# Metode dalam Java



## Pengertian Metode

Metode adalah blok kode yang digunakan untuk melakukan tindakan tertentu dan terletak di dalam kelas. Metode sering disebut juga sebagai fungsi. Metode hanya akan dieksekusi atau berjalan ketika dipanggil.

Sebuah metode harus dideklarasikan di dalam sebuah kelas. Metode didefinisikan dengan nama metode, diikuti oleh tanda kurung ().



# Metode dalam Java



## Deklarasi Metode

*Access Specifier*

*Return Type*

*Method Name*

*Parameter List*

```
public int tambahAngka(int a, int b) {  
    int hasilTambah = a + b;  
    return hasilTambah;  
}
```

*Method Body*



# Metode dalam Java



## Access Specifier

Access Specifier adalah tipe akses dari sebuah metode. Access Specifier menentukan visibilitas dari metode. Java memiliki 4 Access Specifier :

Public

Metode dapat diakses dari manapun baik itu dari class yang berbeda maupun package yang berbeda

Private

Ketika menggunakan private specifier, metode ini hanya dapat diakses di dalam kelas dimana metode itu dideklarasikan

Protected

Ketika menggunakan protected specifier, metode ini hanya dapat diakses di dalam paket yang sama atau sub-kelas di dalam paket yang berbeda

Default

Ketika metode tidak menggunakan access specifier apapun, Java secara otomatis akan menggunakan default specifier. Default specifier hanya dapat diakses oleh package yang sama.



# Metode dalam Java



## Return type

Return type adalah tipe data yang dikembalikan oleh metode. Tipe data yang dikembalikan bisa berupa int, string, double, boolean, long dan lainnya. Jika metode tidak mengembalikan data apapun, gunakan void sebagai return type.

```
public int hitungAngka() {  
    return 20 + 30;  
}
```

```
public double luasLingkaran() {  
    return 3.14 * 13 * 13;  
}
```

```
public long jarakPlanet() {  
    return 15000000000L * 90;  
}
```

```
public String berisalam() {  
    return "Halo Selamat Siang";  
}
```

```
public boolean hidupListrik() {  
    return true;  
}
```

```
public void simpanAngka() {  
    int a = 20;  
}
```



# Metode dalam Java



## Parameter List

Parameter list adalah daftar parameter yang dipisahkan oleh koma dan diapit oleh tanda kurung yang berisi tipe data dan nama variable. Jika metode tidak memiliki parameter, biarkan tanda kurung dalam keadaan kosong.

```
public int tambahAngka(int a, int b) {  
    int hasilTambah = a + b;  
    return hasilTambah;  
}
```

```
public int tambahAngka() {  
    int a = 10;  
    int b = 30;  
    int hasilTambah = a + b;  
    return hasilTambah;  
}
```



# Metode dalam Java



## Jenis Method

- Berdasarkan Sifatnya :
  1. **Void method** : Jenis method yang tidak mengembalikan nilai apa pun. Digunakan untuk melakukan tugas atau operasi tertentu tanpa menghasilkan nilai kembali.
  2. **Non void method** : method yang mengembalikan nilai, digunakan untuk melakukan tugas tertentu dan mengembalikan hasil perhitungan atau data ke pemanggilnya.
- Berdasarkan Parameter :
  1. Tanpa parameter
  2. Ada parameter



# Metode dalam Java



## Jenis Method

1. Method Void / tidak mengembalikan nilai

```
class Kotak {  
    double panjang, lebar, tinggi;  
  
    public void cetakVolume() {  
        System.out.println ("volume kotak = "(panjang * lebar * tinggi));  
    }  
}
```

```
public class Hitung {  
    public static void main(String[] args) {  
        Kotak kotak1 = new Kotak();  
  
        kotak1.panjang = 10;  
        kotak1.lebar = 5;  
        kotak1.tinggi = 3;  
  
        kotak1.cetakVolume();  
    }  
}
```

Output :

```
] --- exec:3.1.0:exec (  
· volume kotak = 150.0  
-----
```





# Metode dalam Java



## Jenis Method

### 2. Method non Void / mengembalikan nilai

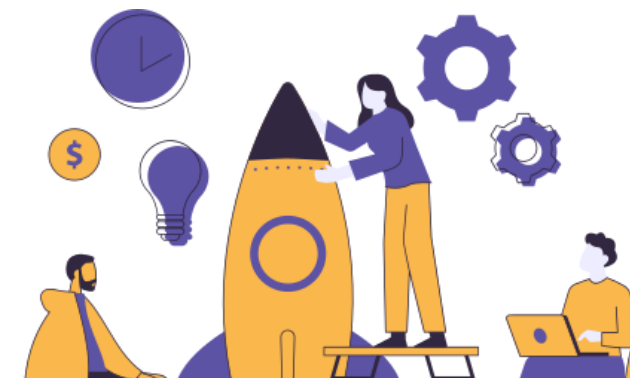
```
class Kotak {  
    double panjang, lebar, tinggi;  
    public double hitungVolume() {  
        double volume = panjang * lebar * tinggi;  
        return volume;  
    }  
}  
  
public class Hitung {  
    public static void main(String[] args) {  
        Kotak kotak1 = new Kotak();  
  
        kotak1.panjang = 10;  
        kotak1.lebar = 5;  
        kotak1.tinggi = 3;  
  
        double vol;  
        vol = kotak1.hitungVolume();  
        System.out.println ("Maka volume kotak adalah "+vol);  
    }  
}
```

Output :

```
--- exec:3.1.0:exec (default-cl  
Maka volume kotak adalah 150.0  
-----
```



# Metode dalam Java



## Jenis Method

### 3. Method berparameter

```
class Kotak {  
  
    public void cetakVolume(double panjang, double lebar, double tinggi){  
        System.out.println ("volume kotak = "+(panjang * lebar * tinggi));  
    }  
}  
  
public class Hitung {  
    public static void main(String[] args) {  
        Kotak kotak1 = new Kotak();  
  
        //kotak1.panjang = 10;  
        //kotak1.lebar = 5;  
        //kotak1.tinggi = 3;  
  
        kotak1.cetakVolume (panjang:15, lebar: 10, tinggi: 2);  
    }  
}
```

Output :

```
--- exec:3.1.0:exec (de  
volume kotak = 300.0  
-----
```

# Contoh

- Menambahkan method ke dalam kelas

```
class Balok {  
    double panjang, lebar, tinggi;  
  
    void volume() {  
        System.out.print(s: "Volume Balok adalah ");  
        System.out.println(panjang * lebar * tinggi);  
    }  
}  
  
public class Hitungbalok1 {  
  
    public static void main(String[] args) {  
        Balok balok1 = new Balok();  
        Balok balok2 = new Balok();  
  
        balok1.panjang = 10;  
        balok1.lebar = 20;  
        balok1.tinggi = 5;  
  
        balok2.panjang = 6;  
        balok2.lebar = 10;  
        balok2.tinggi = 5;  
  
        balok1.volume();  
        balok2.volume();  
    }  
}
```

Output :

Volume Balok adalah 1000.0

Volume Balok adalah 300.0

-----

# Contoh

- Method non void

Output :

```
Volume balok pertama adalah 1000.0
Volume balok kedua adalah 300.0
-----
```

```
class Balok {
    double panjang, lebar, tinggi;

    double volume() {
        return panjang * lebar * tinggi;
    }
}

public class Hitungbalok1 {

    public static void main(String[] args) {
        Balok balok1 = new Balok();
        Balok balok2 = new Balok();
        double vol;

        balok1.panjang = 10;
        balok1.lebar = 20;
        balok1.tinggi = 5;

        balok2.panjang = 6;
        balok2.lebar = 10;
        balok2.tinggi = 5;

        vol = balok1.volume();
        System.out.println ("Volume balok pertama adalah "+vol);
        vol = balok2.volume();
        System.out.println ("Volume balok kedua adalah "+vol);
    }
}
```

# Contoh

- Method Berparameter

Output :

Volume balok pertama adalah 1000.0

Volume balok kedua adalah 300.0

---

```
class Balok {
    double panjang, lebar, tinggi;
    double volume() {
        return panjang * lebar * tinggi;
    }
    void setUkuran (double p, double l, double t) {
        panjang = p;
        lebar = l;
        tinggi = t;
    }
}

public class Hitungbalok1 {
    public static void main(String[] args) {
        Balok balok1 = new Balok();
        Balok balok2 = new Balok();
        double vol;

        balok1.setUkuran(p: 10, l: 20, t: 5);
        balok2.setUkuran(p: 6, l: 10, t: 5);


        vol = balok1.volume();
        System.out.println ("Volume balok pertama adalah "+vol);
        vol = balok2.volume();
        System.out.println ("Volume balok kedua adalah "+vol);
    }
}
```

# Deklarasi Paket dan Paket Import



Paket (package) dalam Java adalah nama untuk sekelompok jenis kelas, interface, dan sub paket yang serupa. Deklarasi paket membantu dalam mengorganisir kode dan menghindari konflik nama antar kelas. Paket di Java dapat dikategorikan dalam dua bentuk, paket built-in dan paket yang dibuat oleh pengguna.

## Jenis – jenis package :

1. Built-in package (paket bawaan java) : Paket-paket yang disediakan oleh bahasa pemrograman Java itu sendiri. Contoh: `java.lang`, `java.util`, `java.io`
  2. Custom package : paket yang dibuat oleh pengembang (user) untuk mengorganisir kelas-kelas yang terkait dalam proyek mereka sendiri
- 

# Deklarasi Paket dan Paket Import

Untuk menggunakan kelas dari paket di dalam sebuah program, paket harus di deklarasikan dengan nama paket dan kelas yang digunakan.

`java.util.Scanner;`

Paket                      Nama kelas

```
public static void main(String[] args) {  
    java.util.Scanner keyboard = new java.util.Scanner(System.in);  
    System.out.print("Masukkan angka : ");  
    int angka = keyboard.nextInt();  
    System.out.println("angka yang anda masukkan adalah : "+angka);  
}
```

Menggunakan nama kelas dengan paketnya seperti di atas mengakibatkan nama yang sangat panjang untuk sebuah kelas. Nama yang panjang akan menurunkan tingkat readability yang membuatnya sulit untuk dibaca

# Deklarasi Paket dan Paket Import




## Import Statement

Dengan menggunakan import statement, penamaan kelas yang panjang bisa dihindari

```
import java.util.Scanner; } Import statement
```

```
public class HelloWorld {  
    public static void main(String[] args) {  
        Scanner keyboard = new Scanner(System.in);  
        System.out.print("Masukkan angka : ");  
        int angka = keyboard.nextInt();  
        System.out.println("angka yang anda masukkan adalah :  
"+angka);  
    }  
}
```





# Deklarasi Paket dan Paket Import



## Menggunakan package :

1. Class yang menggunakan suatu package, berada dalam direktori yang sama dengan class – class yang digunakan sehingga tidak diperlukan import.
2. Class yang menggunakan suatu package, berada dalam direktori yang berbeda dengan class – class yang digunakan, sehingga pada awal source code di class yang menggunakan harus mencantumkan :

`Import namaPackage>NamaClass; atau`

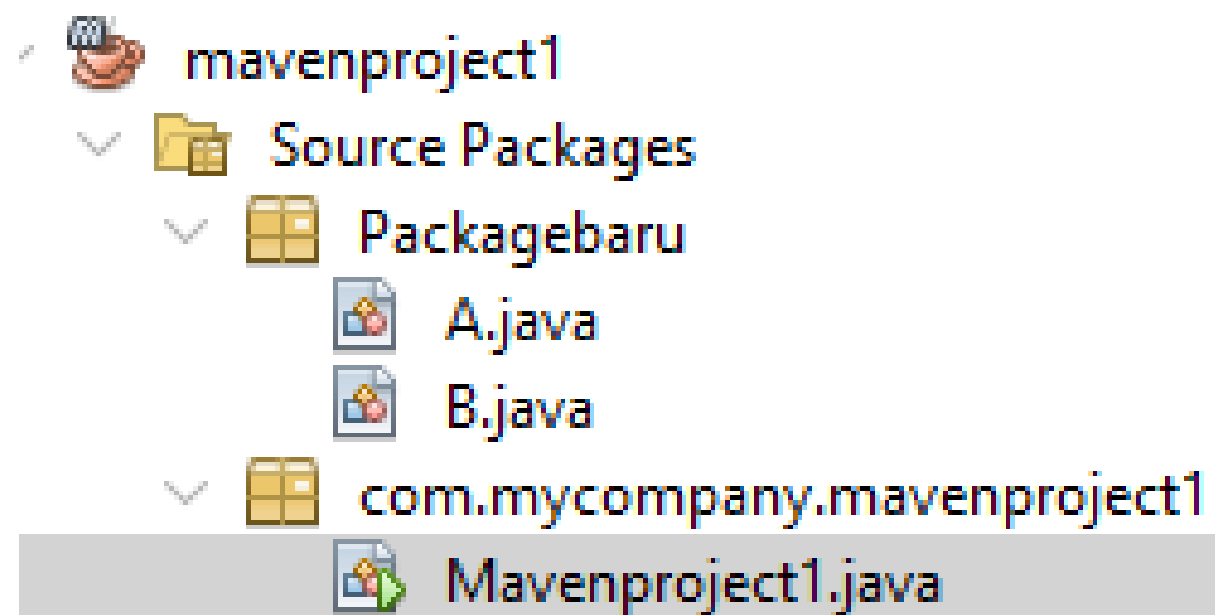
`Import namaPackage.*;`



# Deklarasi Paket dan Paket Import

## Import Statement : Semua Kelas di Dalam Paket

Jika mengimport semua kelas di dalam paket ke dalam program, import statement ditulis nama paket terlebih dahulu, diikuti titik dan tanda asterisk (\*)

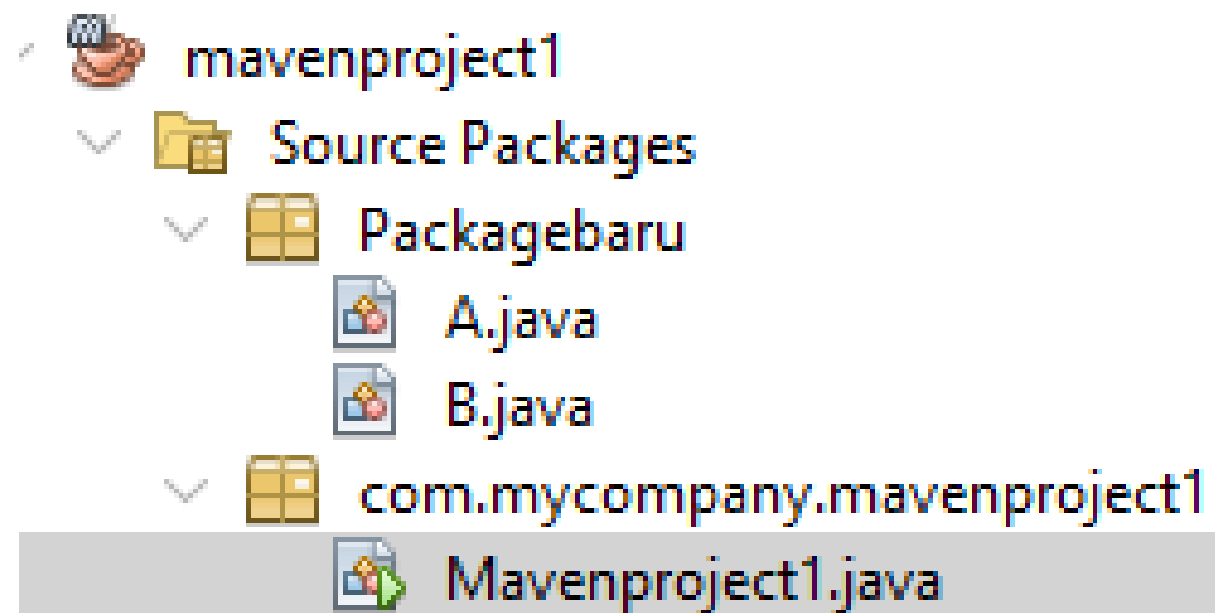


```
package com.mycompany.mavenproject1;  
import Packagebaru.*;  
  
public class Mavenproject1 {  
  
    public static void main(String[] args) {  
        A objek1 = new A();  
        B objek2 = new B();  
  
        objek1.salam();  
        objek1.perkenalan();  
  
        objek2.tanya();  
    }  
}
```

# Deklarasi Paket dan Paket Import

## Import Statement : Single Class

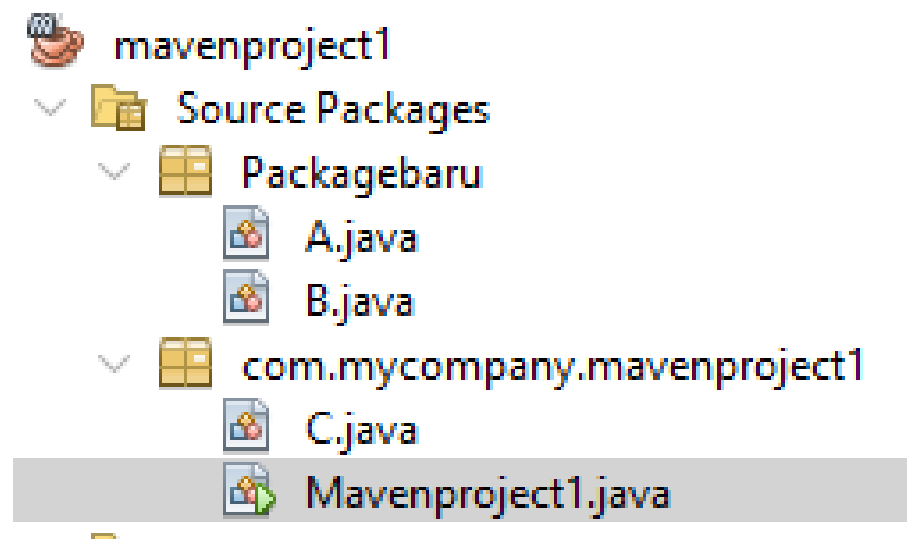
Jika mengimport satu kelas ke dalam program, import statement ditulis nama paket terlebih dahulu, diikuti titik dan nama kelas yang ingin diimport



```
package com.mycompany.mavenproject1;  
import Packagebaru.A;  
  
public class Mavenproject1 {  
  
    public static void main(String[] args) {  
        A objek1 = new A();  
        B objek2 = new B();  
  
        objek1.salam();  
        objek1.perkenalan();  
  
        objek2.tanya();  
    }  
}
```

# Deklarasi Paket dan Paket Import

Mengakses kelas dengan package yang sama :



```
package com.mycompany.mavenproject1;

public class C {
    String nama = "Budi";
    int NIM = 220711;
}
```

```
package com.mycompany.mavenproject1;

public class Mavenproject1 {

    public static void main(String[] args) {
        C data = new C();
        System.out.println("Nama Mahasiswa : "+data.nama);
        System.out.println("NIM : "+data.NIM);
    }
}
```

Output :

```
Nama Mahasiswa : Budi
NIM : 220711
```



# Daftar Pustaka



Herbert Schildt, 2007, *Java™: The Complete Reference, Seventh Edition*,  
McGraw-Hill Companies, United States, 62-73





Thank You

