VOID METHOD
&
NON VOID METHOD

Fundamen Pengembangan Aplikasi

UNIVERSITAS
ISLAM
INDONESIA

METHOD

Fundamen Pengembangan Aplikasi

UNIVERSITAS
ISLAM
INDONESIA

Definisi Method

- **Method** adalah blok program yang akan dijalankan saat dipanggil. Method dapat **menerima data** dan **melakukan beberapa aksi**. Method juga dikenal sebagai fungsi/**function** (w3school.com)
  - **Tujuan**: Untuk menggunakan kode secara **berulang** (mendefinisikan kode sekali dan menggunakannya berkali-kali)

- **Method** adalah kumpulan statement yang melakukan suatu tugas tertentu. Method juga dapat **mengembalikan** suatu nilai ke pemanggilnya (www.geeksforgeeks.org)
  - **Tujuan**: Untuk menghemat waktu (dapat menggunakan suatu perintah tanpa harus menuliskan lagi)

Method: Analogi

- Perhatikan alur proses di samping! Kira-kira apa yang ingin saya buat?
- Mana yang lebih **enak** disebutkan jika Anda ingin seseorang membuatkan kopi untuk Anda?
  - Proses-proses dalam membuat kopi? atau
  - **cukup** menyebutkan "**buatkan kopi**"?
- Pada contoh ini, **Buat Kopi** adalah suatu **method**, di mana terdapat ***proses-proses pembuatannya*** yang dapat dianalogikan sebagai ***statement***

Method, Fungi, dan Prosedur
(Hubungan dan Komparasi)

- Pada pemrograman Java, **method** adalah blok program terpisah (**di luar blok program utama**) yang dapat digunakan untuk menyelesaikan masalah khusus
- **Fungsi** adalah method yang mengembalikan suatu nilai (juga disebut sebagai **Non Void Method**)
  - Beberapa sumber juga menyebut Non Void Method dengan istilah **Return Method**
- **Prosedur** adalah method yang **TIDAK** mengembalikan suatu nilai (juga disebut sebagai **Void Method**)

STRUKTUR METHOD

Fundamen Pengembangan Aplikasi

UNIVERSITAS
ISLAM
INDONESIA

Struktur Method Void

```
static void <namaMethod> (<parameter>){
    <statement>
}
```

- **void**
  - keyword yang menandakan bahwa method yang dibuat merupakan method void **tanpa nilai kembalian** → **Prosedur**
- **namaMethod**
  - nama method (spesifik, jelas, dan sesuai dengan aturan penamaan)
- **parameter**
  - parameter yang digunakan pada method (***opsional***)

Contoh Method Void

Contoh:

```
public class MethodVoid {
    //pembuatan method void
    static void sapa(){
        System.out.println("Halo Dunia");
    }

    public static void main(String[] args) {
        //pemanggilan method void
        sapa();
    }
}
```

Void Method

Main Method

*Di dalam **class** yang ada ***main method*** maka method void diletakkan **di luar** main method, baik di atasnya atau di bawahnya.

Method Void dengan Parameter

Contoh:

```
public class MethodVoid {
    //pembuatan method void
    static void sapa(String nama){
        System.out.println("Halo, "+nama);
    }

    public static void main(String[] args) {
        //pemanggilan method void
        sapa("Agus");
    }
}
```

Parameter Formal

Parameter Aktual

Struktur Method Non Void

```
static <tipeData> <namaMethod> (<parameter>){
        <statement>
        return <nilaiBalik>
}
```

- **tipeData** : tipe data dari nilai yang dikembalikan oleh method
- **namaMethod** : nama method (spesifik, jelas, dan sesuai dengan aturan penamaan)
- **parameter** : parameter yang digunakan pada method (opsional)
- **nilaiBalik**: nilai yang dikembalikan **sesuai** dengan tipe data yang didefinisikan di method-nya
  - Nilai balik akan ***dikembalikan*** ke kode program yang memanggil method ini, entah langsung ***ditampilkan***, ***disimpan*** ke dalam variabel, atau ***digunakan*** oleh method lainnya

Struktur Method Non Void

```
static <tipeData> <namaMethod> (<parameter>){
    <statement>
    return <nilaiBalik>
}
```

<tipeData>

- Bisa diisi ***tipe data primitif*** sesuai dengan **<nilaiBalik>** (boolean, byte, short, int, long, float, double, atau char)
- Bisa diisi ***tipe data reference***/bentukan sesuai dengan nilai balik (String, ArrayList, dll)
- Bisa diisi dengan kata "**void**" jika method **tidak memiliki nilai balik** (procedure)

Struktur Method Non Void

```
static int <namaMethod> (<parameter>){
    <statement>
    return <nilaiBalik>
}
```

<namaMethod>

- Dapat menggambarkan/mendeskripsikan method
- Bisa terdiri dari huruf, angka, dan garis bawah (_)
- Karakter pertama tidak boleh angka atau simbol operator
- Tidak boleh mengandung spasi atau operator
- Bisa menggunakan teknik **CamelCase** untuk menggantikan spasi

Struktur Method Non Void

```
static int LuasPersegiPanjang (<parameter>){
    <statement>
    return <nilaiBalik>
}
```

<parameter>

- ***Parameter*** adalah informasi yang **dikirimkan** pada **method** dan berperan sebagai ***variabel*** pada method (parameter formal)
- Format penulisan parameter: **<tipeData> <namaParameter>**
- Tipe data dapat terdiri dari tipe data primitif, reference, atau tipe data bentukan.
- Parameter bersifat **opsional** (bisa tidak ada, bisa ada 1, bisa lebih dari 1)
- Untuk parameter lebih dari 1, harus ***dipisahkan dengan koma***

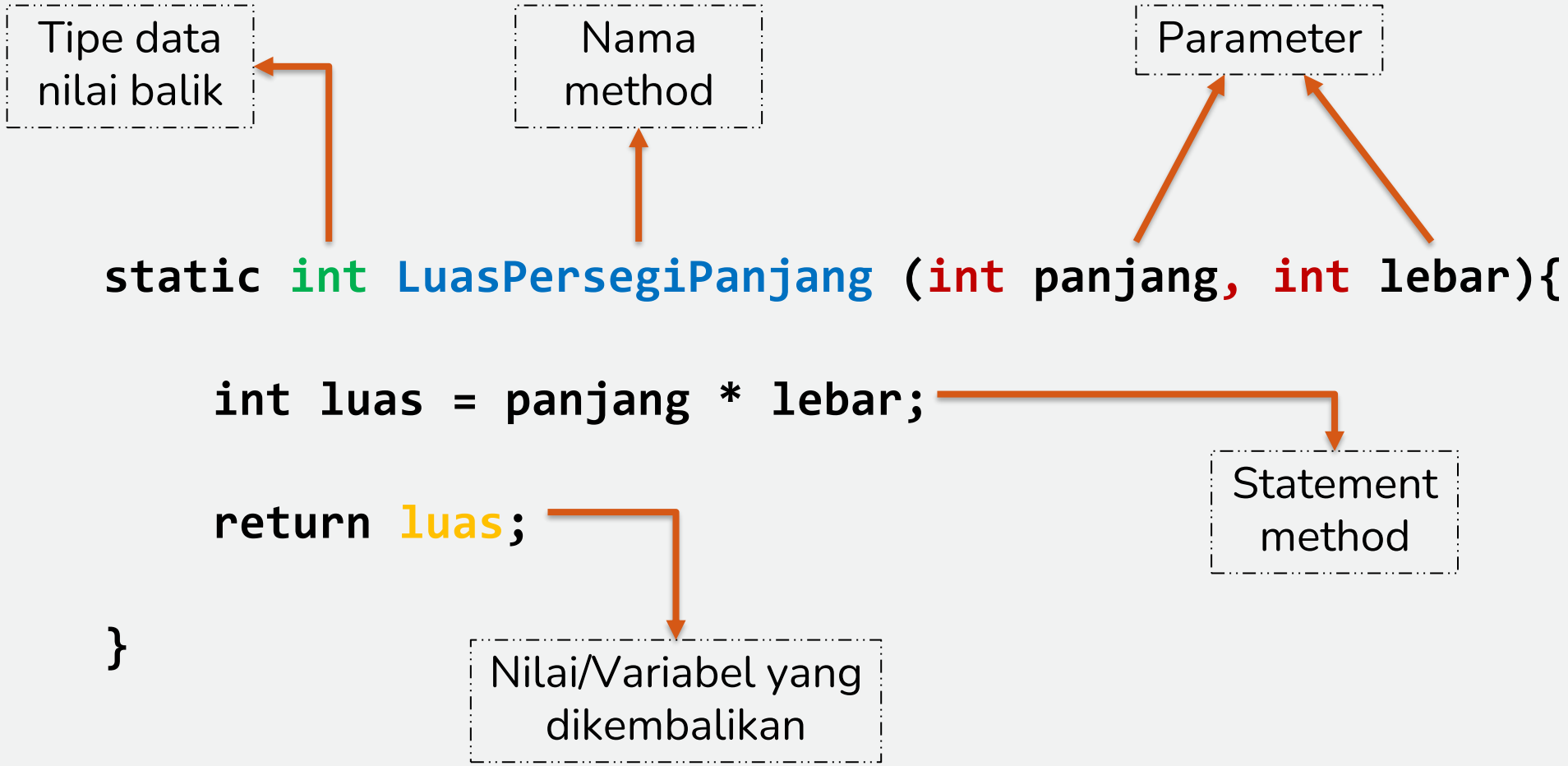
Struktur Method Non Void

```
static int LuasPersegiPanjang (int panjang, int lebar){
    int luas = panjang * lebar;
    return <nilaiBalik>
}
```

<nilaiBalik>

- Nilai balik adalah nilai akhir yang ***dikirimkan*** oleh sebuah method (layaknya sebuah fungsi)
- Penulisan nilai balik **harus** diawali dengan menuliskan **return**
- Tiap Non Void Method **hanya** dapat memberikan **satu nilai balik**
- **Nilai** atau **variabel** yang dikembalikan **HARUS** memiliki tipe data yang **SAMA** dengan tipe data yang **didefinisikan** di method

Contoh Method Non Void



Contoh Pemanggilan Method Non Void

```
public class FungsiLuas {
    public static void main(String[] args) {
        int luasPS = LuasPersegiPanjang(10, 5);
        System.out.println("Luas: "+luasPS);
    }

    static int LuasPersegiPanjang (int panjang, int lebar){
        int luas = panjang * lebar;
        return luas;
 }
}
```

Pemanggilan fungsi dengan parameter

Perbedaan Method Void dan Non Void

Void

1. Menggunakan keyword **void** ketika membuat method
2. **TIDAK** menggunakan keyword **tipe data** ketika membuat method
3. **TIDAK** terdapat statemen **return** di dalam isi method-nya

Non Void

1. **TIDAK** menggunakan keyword **void** ketika membuat method
2. **Keyword tipe data** harus didefinikan ketika membuat method untuk menunjukkan tipe data dari nilai yang **dikembalikan**
3. Ada statemen **return** di dalam isi method-nya

Perbedaan Method Void dan Non Void

```
static void <namaMethod> (<parameter>){
      <statement>
}
```

→ Void

```
static <tipeData> <namaMethod> (<parameter>){
      <statement>
      return <nilaiBalik>
}
```

→ Non Void

Pemanggilan Method

- **Dituliskan nama methodnya** dan **diikuti** dengan **argumen** (nilai parameter) **jika** method tersebut menggunakan parameter

```
static void menyapa(String nama){
    System.out.println("Halo " + nama);
}

public static void main(String[] args) {
    menyapa("Mahasiswa");
}
```

- **Perhatikan** bahwa variabel **nama** memiliki tipe data **String** sehingga ketika pemanggilannya **harus** menggunakan data/nilai yang juga String

- Nilai **"Mahasiswa"** menjadi **parameter aktual** ketika pemanggilan fungsi
- Lalu akan **mengisi** variabel **nama** (pada fungsi) untuk dieksekusi

Pemanggilan method harus sesuai dengan namanya (***case sensitive***)

Pemanggilan Method

- **Non void method** → karena menggunakan ***nilai balik*** maka **ketika dipanggil tidak dapat berdiri sendiri**, sehingga **harus**:
  - **Ditampilkan langsung (output)**
  - Disimpan ke dalam variabel
  - Digunakan oleh fungsi/prosedur lainnya

```
static int kuadrat(int n){
    return n*n;
}

public static void main(String[] args) {
    System.out.println(kuadrat(5));
}
```

Nilai yang dikembalikan ***langsung ditampilkan***

Pemanggilan Method

- **Non void method** → karena menggunakan ***nilai balik*** maka **ketika dipanggil tidak dapat berdiri sendiri**, sehingga **harus**:
  - Ditampilkan langsung (output)
  - **Disimpan ke dalam variabel**
  - Digunakan oleh fungsi/prosedur lainnya

```
static int kuadrat(int n){
    return n*n;
}

public static void main(String[] args) {
    int hasilKuadrat = kuadrat(5);
}
```

Nilai yang dikembalikan ***disimpan ke dalam variabel***

Pemanggilan Method

- **Non void method** → karena menggunakan ***nilai balik*** maka **ketika dipanggil tidak dapat berdiri sendiri**, sehingga **harus**:
  - Ditampilkan langsung (output)
  - **Ditampung ke dalam variabel**
  - **Digunakan oleh fungsi/prosedur lainnya**

```
static int kuadrat(int n){
    return n*n;
}

public static void main(String[] args) {
    cetakHasil(kuadrat(5));
}
```

Nilai yang dikembalikan ***digunakan oleh fungsi/prosedur lainnya***

CONTOH

Fundamen Pengembangan Aplikasi

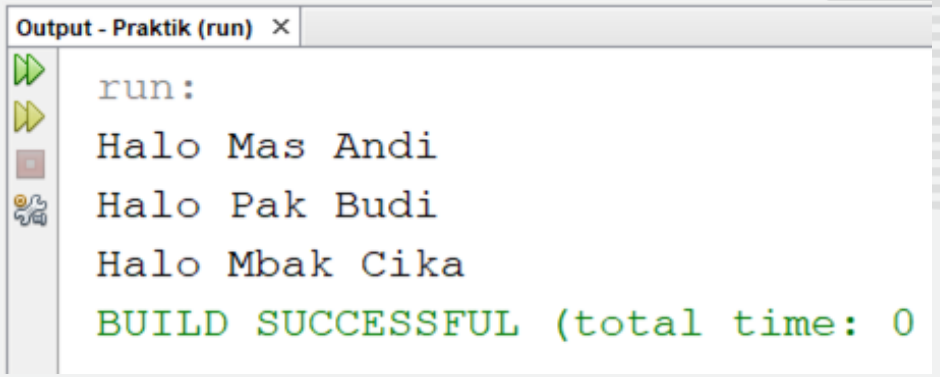
UNIVERSITAS
ISLAM
INDONESIA

Contoh 1

- Membuat method untuk menampilkan teks dengan format sebagai berikut:

  Halo <panggilan> <nama>
- Terdapat **2 parameter**
  - **Parameter pertama** mendefinisikan panggilan dengan menggunakan **kode** dalam bentuk **bilangan bulat** dengan aturan sebagai berikut

    1: Mas; 2: Mbak; 3: Pak; 4: Bu
  - **Parameter kedua** mendefinisikan **nama**
- Akan muncul seperti gambar di samping jika dimasukkan argument berikut:



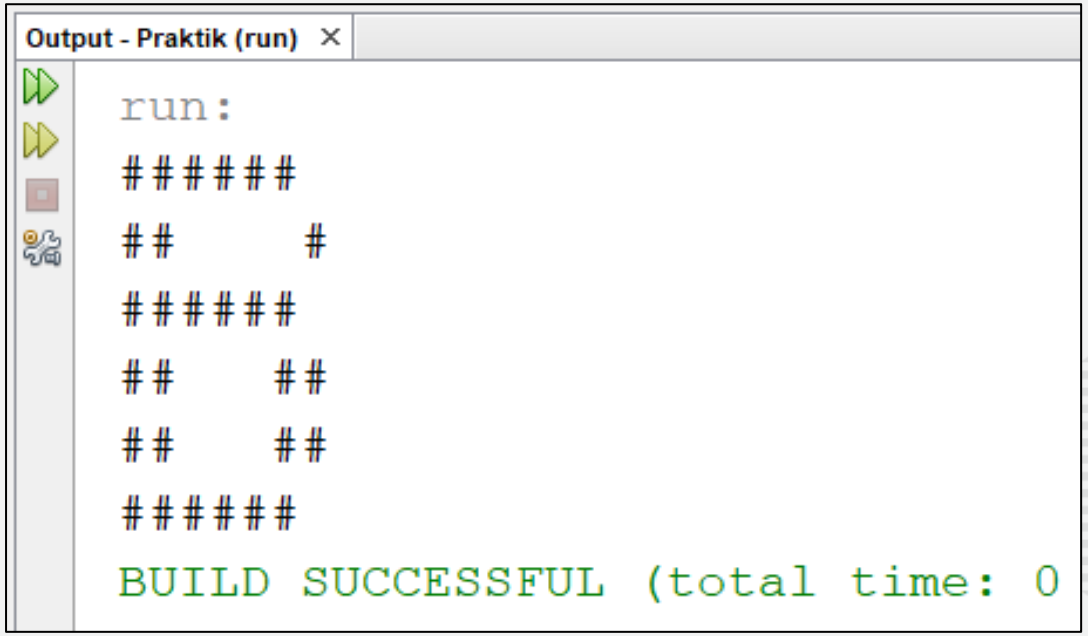
Contoh 2

- Membuat method untuk menggambar karakter pagar (#) dan spasi
- Terdapat 3 parameter
  1. Parameter **pertama** mendefinisikan berapa banyak karakter pagar akan ditulis
  2. Parameter **kedua** mendefinisikan berapa banyak karakter spasi akan ditulis setelah karakter pagar
  3. Parameter **ketiga** mendefinisikan berapa banyak karakter pagar akan ditulis setelah karakter spasi
- Misal saya memberikan argumen (3, 2, 3), maka akan menampilkan:
  ### ###
- Misal saya memberikan argumen (3, 0, 3), maka akan menampilkan:
  ######

Contoh 2

- Misal saya memasukkan kombinasi dari argumen-argumen berikut:

Parameter	Ke-1	Ke-2	Ke-3
Argumen	6	0	0
	2	4	1
	6	0	0
	2	3	2
	2	3	2
	0	0	6

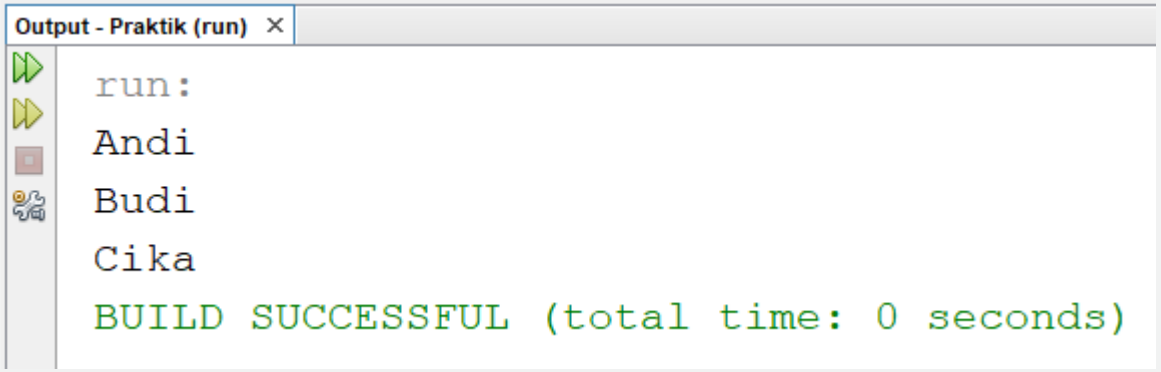


Contoh 3

- Membuat method yang menghasilkan nilai balik berupa **total penjumlahan** dari **3 bilangan bulat**
- Terdapat 3 parameter
  - Parameter **pertama** mendefinisikan bilangan pertama
  - Parameter **kedua** mendefinisikan bilangan kedua
  - Parameter **ketiga** mendefinisikan bilangan ketiga
- Nilai balik kemudian kemudian **disimpan** dalam variabel bernama **total**
- Variabel **total** kemudian **ditampilkan** sebagai keluaran
- Jika saya memasukkan argumen dengan nilai (5, 2, 3), keluaran yang muncul adalah 10
- Jika saya memasukkan argumen dengan nilai (-7, -8, -9), keluaran yang muncul adalah -24

Contoh 4

- Membuat method **tanpa nilai balik** yang dapat **menampilkan** nilai elemen suatu larik secara beruntun
- Hanya menggunakan **1 parameter** yang mendefinisikan larik yang akan ditampilkan
- Gunakan perintah **<namaLarik>.length** untuk mengetahui panjang larik
- Jika saya memiliki larik yang berisi elemen “Andi”, “Budi”, dan “Cika”, keluaran yang dihasilkan adalah sebagai berikut



TERIMA KASIH

Fundamen Pengembangan Aplikasi

UNIVERSITAS
ISLAM
INDONESIA