

# METHOD

# Outline Materi

- Definisi Method
- Deklarasi Method
- Pemanggilan Method
- Lingkup variabel
- *Passing by value*
- *Passing arrays (passing by reference)*
- *Return arrays*
- *Overloading Method*

# Analogi Method

- ❑ Amir, Budi dan Cici mendapat tugas oleh Ibu mereka untuk membersihkan rumah.
- ❑ Si Ibu kemudian membagi tugas tersebut agar lebih mudah dan cepat diselesaikan.
- ❑ Amir mendapat tugas mengepel lantai. Budi bertugas mengatur barang-barang di dalam rumah. Cici bertugas mengecat dinding rumah.
- ❑ Si Ibu dapat dengan mudah mengetahui sumber kesalahan apabila terjadi.

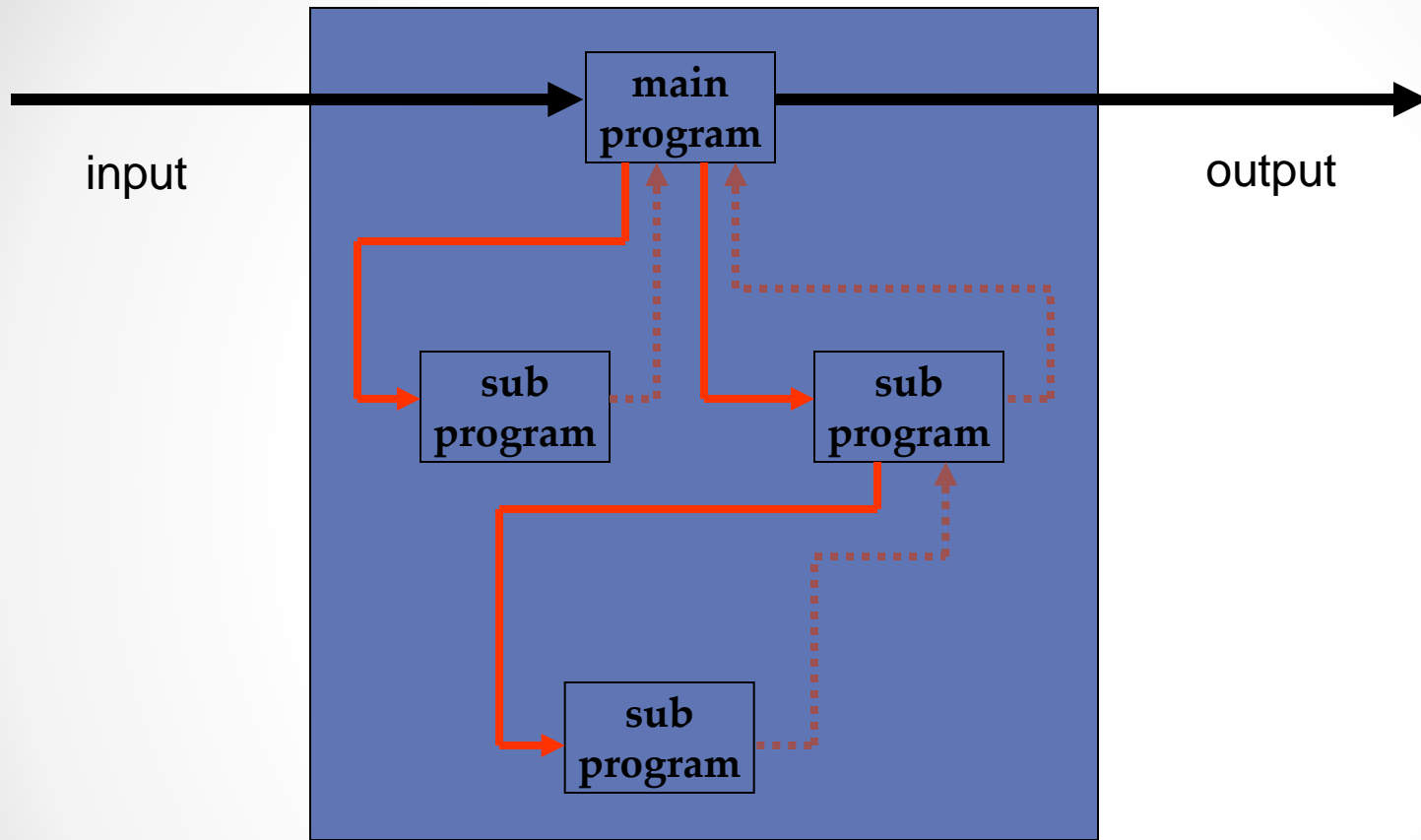
# Analogi Method

- ❑ Apabila lantai masih terlihat kotor, tentunya kesalahan tersebut dilakukan oleh Amir.
- ❑ Apabila barang-barang masih terlihat berantakan, maka Budi-lah biang keroknya.
- ❑ Sedangkan bila dinding terlihat masih kusam, maka tentunya Cici yang melakukan kesalahan.
- ❑ Ibu dapat dengan mudah mengontrol kegiatan tersebut hingga selesai dengan benar.

Si Ibu dapat kita anggap sebagai diri kita sendiri (programmer), sedangkan pekerjaan yang ingin diselesaikan adalah program yang ingin kita buat.

Amir, Budi, dan Cici dapat dianggap sebagai Method dimana setiap method diberikan tugas yang tertentu sehingga bila terjadi kesalahan dalam logika pemrograman, maka dapat dengan mudah dilacak letak kesalahannya untuk kemudian diperbaiki.

# Gambaran Penggunaan Method



proses (program)

Keterangan:

→ method call (pemanggil fungsi),  
memberikan input ke fungsi

→ return (kembalian methodi),  
mengeluarkan output

# Definisi Method

- Kumpulan statement yang dikelompokkan bersama untuk suatu operasi
- Contoh:
  - `println()` pada `System.out`
  - `showMessageDialog()/showInputDialog()` pada `JOptionPane`
  - `nextInt()` pada `Scanner`
  - `equals()` pada `String`
- Di dalam method `println()` dari `System.out`
  - Terdiri dari sekumpulan statements
  - Bertujuan mencetak output ke console

# Deklarasi Method

- Sintaks:  
modifier returnValueType methodName(list of parameters)  
{  
    statements;  
}
- Modifier
  - Status/sifat dari method, cara bagaimana method dapat dipanggil
  - Contoh: public, private, static
  - Optional
  - Akan dijelaskan lebih lanjut di PBO
- returnValueType
  - Nilai yang dikembalikan dari method
  - Berupa tipe data primitif, String, atau Array
  - Optional, jika tidak mengembalikan nilai maka **void**
  - Jika mengembalikan nilai maka memerlukan keyword **return**



# Deklarasi Method

- **methodName**
  - Nama method
  - Sebaiknya mengikuti konvensi penamaan method di Java
- **List of parameters**
  - Nilai yang dikirimkan ke method
  - Berupa tipe data primitif, String, atau Array
  - Optional, jika tidak menggunakan parameter maka dikosongkan
  - Parameter yang dideklarasikan → *formal/simply parameters*
  - Parameter yang dikirim → *actual parameters*
- **Contoh:**

```
public static void cetak10bintang()  
{  
    for(int i=0; i<10; i++)  
        System.out.println("*");  
}
```

# Deklarasi Method

**modifier**   **return value**   **method name**   **formal parameter**

```
public static int max (int num1, int num2) {  
    int result;  
    if(num1>num2)  
        result = num1;  
    else  
        result = num2;  
    return result;  
}
```

**parameter list**

**method body**

**return value**

```
int z = max(7, 10); //pemanggilan method
```

**actual parameters (arguments)**

# Pemanggilan Method

- Jika method memiliki return value maka pemanggil perlu menampung nilainya
  - Contoh: `bilangan = input.nextInt();`
- Jika method memiliki parameter maka pemanggil perlu mengirim nilai
  - Contoh: `System.out.println("Welcome to Java!");`
- Kombinasi dari return value dan parameter
  - Contoh: `bilangan = Integer.parseInt(kalimat);`
- Method dapat dipanggil dari main ataupun method lainnya

# Pemanggilan Method

```
public class TestMethod1
{
    public static void cetak10bintang()
    {
        for(int i=0; i<10; i++)
            System.out.println("*");
    }

    public static void main(String[] args)
    {
        System.out.println("List 10 bintang");
        cetak10bintang();
        System.out.println("Terima kasih");
    }
}
```

---

```
List 10 bintang
*
*
*
*
*
*
*
*
*
*
Terima kasih
```

# Pemanggilan Method

```
public class TestMethod2
{
    public static void cetakSegitigaBintang(int n)
    {
        for(int i=0; i<n; i++)
        {
            for(int j=0; j<=i; j++)
                System.out.print("*");
            System.out.println();
        }
    }

    public static void main(String[] args)
    {
        System.out.println("Segitga bintang bernilai 4");
        cetakSegitigaBintang(4);
        System.out.println("Segitga bintang bernilai 10");
        cetakSegitigaBintang(10);
    }
}
```

```
Segitga bintang bernilai 4
*
**
***
****
Segitga bintang bernilai 10
*
**
***
****
*****
*****
*****
*****
*****
*****
*****
```

# Pemanggilan Method

```
public class TestMethod3
{
    public static int max(int bil1, int bil2)
    {
        int terbesar;

        if(bil1>bil2)
            terbesar = bil1;
        else
            terbesar = bil2;

        return terbesar;
    }

    public static void main(String[] args)
    {
        int nilai;

        nilai = max(3,5);
        System.out.println("Bilangan terbesar antara 3 dan 5 adalah "+nilai);

        nilai = max(14,7);
        System.out.println("Bilangan terbesar antara 14 dan 7 adalah "+nilai);
    }
}
```

Bilangan terbesar antara 3 dan 5 adalah 5  
Bilangan terbesar antara 14 dan 7 adalah 14

# Pemanggilan Method

- Pada method max

```
public static int max(int bil1, int bil2)
{
    int terbesar;

    if (bil1 > bil2)
        terbesar = bil1;
    else
        terbesar = bil2;

    return terbesar;
}
```

Dapat diganti menjadi:

```
public static int max(int bil1, int bil2)
{
    if (bil1 > bil2)
        return bil1;
    else
        return bil2;
}
```

# Pemanggilan Method

- Pada main

```
public static void main(String[] args)
{
    int nilai;

    nilai = max(3,5);
    System.out.println("Bilangan terbesar antara 3 dan 5 adalah
    "+nilai);

    nilai = max(14,7);
    System.out.println("Bilangan terbesar antara 14 dan 7 adalah
    "+nilai);
}
```

Dapat diganti menjadi

```
public static void main(String[] args)
{
    System.out.println("Bilangan terbesar antara 3 dan 5 adalah
    "+max(3,5));
    System.out.println("Bilangan terbesar antara 14 dan 7 adalah
    "+max(14,7));
}
```



# Lingkup Variabel

- Variabel yang dideklarasikan di suatu method hanya bisa dipakai di method itu
  - Pada TestMethod3, variabel int terbesar hanya dapat dipakai method max(...), tidak dapat dipakai di method main(...)  
Variabel int nilai hanya dapat dipakai di method main(...), tidak dapat dipakai di method max(...)
  - Pada TestMethod2, variabel int n hanya dapat dipakai di method cetakSegitigaBintang(...)  
Variabel i dan j hanya dapat dipakai di dalam lingkup perulangan saat itu saja
- Deklarasi variabel dalam method → variabel lokal (*local variable*)

# Lingkup Variabel

```
public static void cetakSegitigaBintang(int n)
{
    for(int i=0; i<n; i++)
    {
        for(int j=0; j<=i; j++)
        {
            System.out.print("*");
            System.out.println();
        }
    }
}
```

Lingkup variabel n

Lingkup variabel i

Lingkup variabel j

# Lingkup Variabel

- Variabel yang dapat dikenal oleh semua method → variabel global (*global variable*)
- Dideklarasikan di luar method

```
public class TestMethod3
{
    int variabelGlobal;

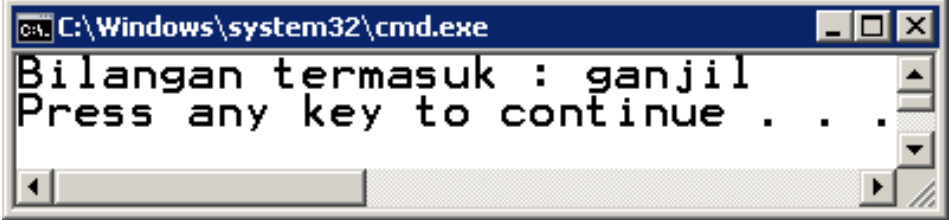
    public static int max(int bil1, int bil2)
    {
        ...
    }

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



Lingkup variabel global

- `public class ganjilgenap`
- `{`
- `public static String gjgn(int bil)`
- `{`
- `String hasil;`
- `if(bil%2==0)`
- `hasil="genap";`
- `else`
- `hasil="ganjil";`
- `return hasil;`
- `}`
- `public static void main(String arg[])`
- `{`
- `System.out.println("Bilangan termasuk : "+gjgn(5));`
- `}`
- `}`



A screenshot of a Windows command prompt window. The title bar reads "C:\Windows\system32\cmd.exe". The window contains the text "Bilangan termasuk : ganjil" on the first line and "Press any key to continue . . ." on the second line. A horizontal scrollbar is visible at the bottom of the text area.

# *Passing by Value*

- Keunggulan method → kemampuan parameter
- Pengiriman parameter berupa nilai (*passing by value*)
- Nilai variabel method pengirim tetap tidak berubah walaupun di method penerima diubah

# Passing by Value

```
public class PassingByValue
{
    public static void tukar(int n1, int n2)
    {
        int nilai = n1;
        n1=n2;
        n2=nilai;
    }

    public static void main(String[] args)
    {
        int bil1=7, bil2=5;

        System.out.println("Nilai bil1="+bil1+" dan bil2="+bil2+" sebelum ditukar");
        tukar(bil1,bil2);
        System.out.println("Nilai bil1="+bil1+" dan bil2="+bil2+" sesudah ditukar");
    }
}
```

Nilai bil1=7 dan bil2=5 sebelum ditukar  
Nilai bil1=7 dan bil2=5 sesudah ditukar

# *Passing Arrays*

- Array dapat dikirim melalui method
- Metode pengiriman → *passing by reference*
- Pengubahan nilai variabel di method penerima mempengaruhi nilai variabel di method pengirim
- Tipe data method pengirim dan penerima harus sama
- Ukuran array penerima akan sama dengan pengirim secara otomatis

# Passing Array

```
public class PassingArray
{
    public static void tukar(int[] n, int idx1, int idx2)
    {
        int nilai = n[idx1];
        n[idx1] = n[idx2];
        n[idx2] = nilai;
    }

    public static void main(String[] args)
    {
        int[] bil = {7, 5};

        System.out.println("Nilai bil[0]="+bil[0]+" dan bil[1]="+bil[1]+" sebelum ditukar");
        tukar(bil,0,1);
        System.out.println("Nilai bil[0]="+bil[0]+" dan bil[1]="+bil[1]+" sesudah ditukar");
    }
}
```

Nilai bil[0]=7 dan bil[1]=5 sebelum ditukar  
Nilai bil[0]=5 dan bil[1]=7 sesudah ditukar



# *Return Arrays*

- *Return value* dapat berupa *array*
- Tipe data *array* penerima dan yang di-*return* harus sama
- *Array* penerima tidak perlu diinisialisasi
- Ukuran *array* penerima akan sama dengan yang di-*return* secara otomatis

# *Return Arrays*

```
public class ReturnArrays
{
    public static int[] bilanganGanjil(int jml)
    {
        int[] hasil = new int[jml];

        for(int i=0; i<jml; i++)
            hasil[i] = (i*2)+1;

        return hasil;
    }

    public static void main(String[] args)
    {
        int[] bilGanjil;

        bilGanjil=bilanganGanjil(10);
        System.out.print("10 bilangan ganjil pertama : ");

        for(int bil : bilGanjil)
            System.out.print(bil+" ");
    }
}
```

10 bilangan ganjil pertama : 1 3 5 7 9 11 13 15 17 19

# Did You Know?

- Dalam beberapa istilah bahasa pemrograman
  - Method yang mengembalikan nilai (*nonvoid*) → *function*
  - Method yang tidak mengembalikan nilai (*void*) → *procedure*
- Passing arrays dapat dilakukan langsung

Contoh:

```
public static void printArray(int[] array)
{
    for(int i=0; i<array.length; i++)
        System.out.print(array[i]+" ");
}
```

Cara pengiriman:

```
printArray(new int[] {3,1, 2, 6, 4, 2});
```

```
public class PassingArraysDirectly
{
    public static void printArray(int[] array)
    {
        for(int i=0; i<array.length; i++)
            System.out.print(array[i]+" ");
    }

    public static void main(String[] args)
    {
        printArray(new int[] {3, 1, 2, 6, 4, 2});
    }
}
```

---

**3 1 2 6 4 2**

# Advanced Learning

- Dua method atau lebih dengan nama yang sama  
→ *method overloading*
- Perbedaan:
  - Tipe data parameter
  - Jumlah parameter
  - Return value
- Contoh:

```
public static int max(int num1, int num2)
public static double max(double num1, double num2)
public static double max(double num1, double num2, double num3)
```

# Advanced Learning

```
public class TestMethodOverloading
{
    public static int max(int num1, int num2)
    {
        return (num1>num2)?num1:num2;
    }

    public static double max(double num1, double num2)
    {
        return (num1>num2)?num1:num2;
    }

    public static double max(double num1, double num2, double num3)
    {
        return max(max(num1,num2),num3);
    }

    public static void main(String[] args)
    {
        System.out.println("The maximum between 3 and 4 is "+max(3,4));
        System.out.println("The maximum between 3.0 and 5.4 is "+max(3.0,5.4));
        System.out.println("The maximum between 3.0, 5.4, and 10.14 is "+max(3.0,5.4,10.14));
    }
}
```

```
The maximum between 3 and 4 is 4
The maximum between 3.0 and 5.4 is 5.4
The maximum between 3.0, 5.4, and 10.14 is 10.14
```

# Advanced Learning

- Overloading methods membuat program lebih jelas dan mudah dibaca
- Overloading methods harus berbeda parameter, tidak dapat hanya berbeda modifiers atau tipe data return value
- Ketidaktercapaian kompilar menentukan method yang akan digunakan → *ambiguous invocation*
- Contoh:

```
public static double max(int num1, double num2)  
public static double max(double num1, int num2)
```

Jika dipanggil dengan → **max(1,2);**  
maka kompilar tidak dapat menentukan method yang akan digunakan

# Latihan

- Buat method untuk menukarkan dua buah nilai yang dimasukkan oleh user.



