Kodehive

Java

Java Fundamental | Bootcamp
OOP III - Function & Polymorphism

Subjects : OOP III

1. Method
2. Method - Input
3. Method - Output
4. Method – Input Output
5. Polymorphism
6. Polymorphism – Overriding
7. Polymorphism - Overloading

Method

- Apa itu method?
- Method adalah sekumpulan perintah yang dikelompokkan dengan nama dan tipe akses tertentu di Java.
- Method sendiri tidak terlepas dengan namanya Procedure (Prosedur) dan Function (Fungsi)

Method

- Kelas tanpa method satupun.

```
public class KelasTanpaMethod {

}
```

Method

- Kelas dengan 1 buah method, namanya main

```
public class KelasDenganMethodMain {

    Run | Debug
    public static void main(String[] args) {
        // TODO Auto-generated method stub

    }

}
```

Method

- Kelas dengan beberapa method

```
public class KelasDenganBeberapaMethod {

    Run | Debug
    public static void main(String[] args) {
        // TODO Auto-generated method stub

    }

    public void metodeSatu() {

    }

    public void metodeDua() {

    }

    public void metodeTiga() {

    }

}
```

Method - Void

- Method void itu disebut juga dengan Procedure (Procedure).
- Void (bahasa inggris) dalam bahasa indonesia artinya kosong. Maksud harfiahnya, method tipe void artinya tidak punya output.

```
public class KelasProcedure {

    public void methodVoid() {
        // ini adalah sebuah methode bernama methodVoid
        // karena tipenya void
    }

}
```

Method – Procedure (Void)

- KelasProcedureLanjut berisi sebuah method namanya methodCetak.
- methodCetak berisi 4 baris kode dari line 6 hingga 9.
- Meski ada kode sysout (cetak) tapi methodCetak tidak menghasilkan sebuah nilai (output), makanya methodCetak disebut dengan sebagai sebuah Procedure.
- Lalu apakaa yang dimaksud dengan method yang memiliki output?

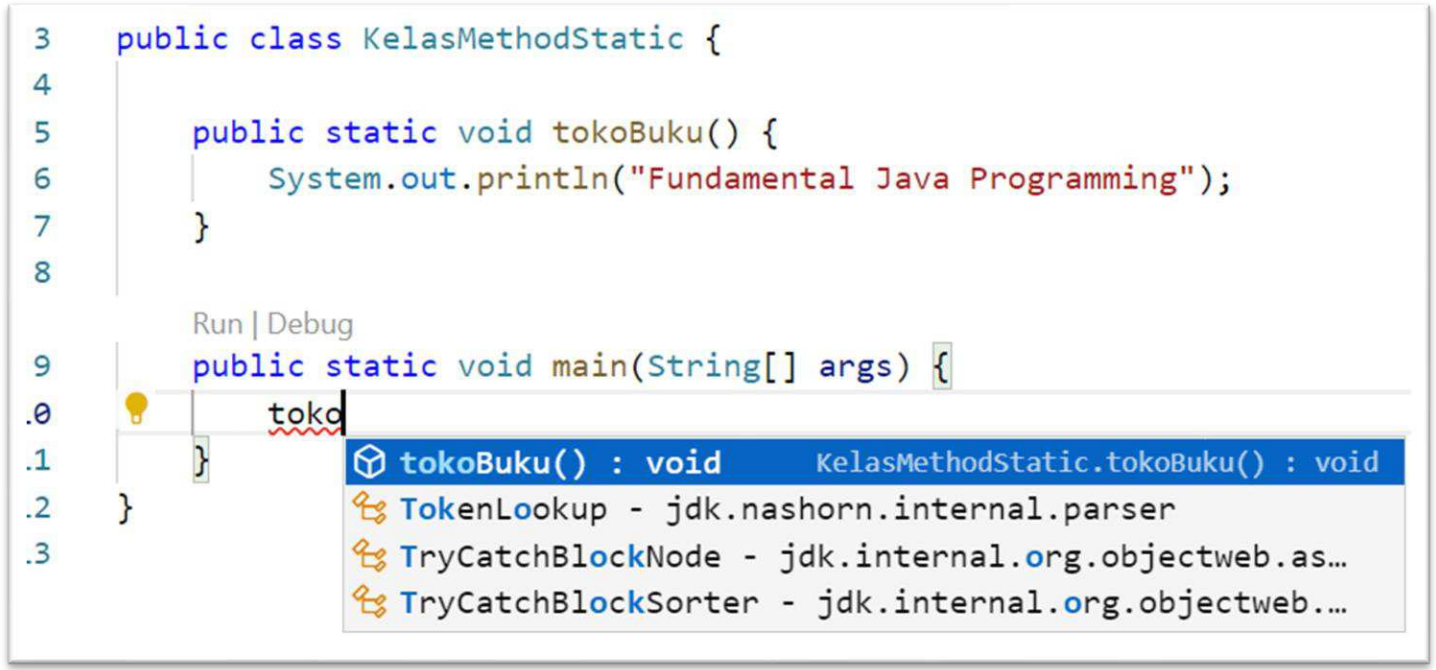
```
public class KelasProcedureLanjut {

    public void methodCetak() {
        int angka = 99;
        String kata = "MANUSIA";
        System.out.println(kata);
        System.out.println(angka);
    }

}
```

Method – Procedure vs Function

- Sekali lagi poinnya adalah sekumpulan sintax dalam suatu label di dalam konsep OOP atau program disebut dengan Method.
- Method yang tidak menghasilkan suatu nilai output disebut dengan Procedure. Cirinya ada **void** nya.
- Sedangkan method yang menghasilkan nilai output disebut dengan Function. Cirinya ada **return** nya.

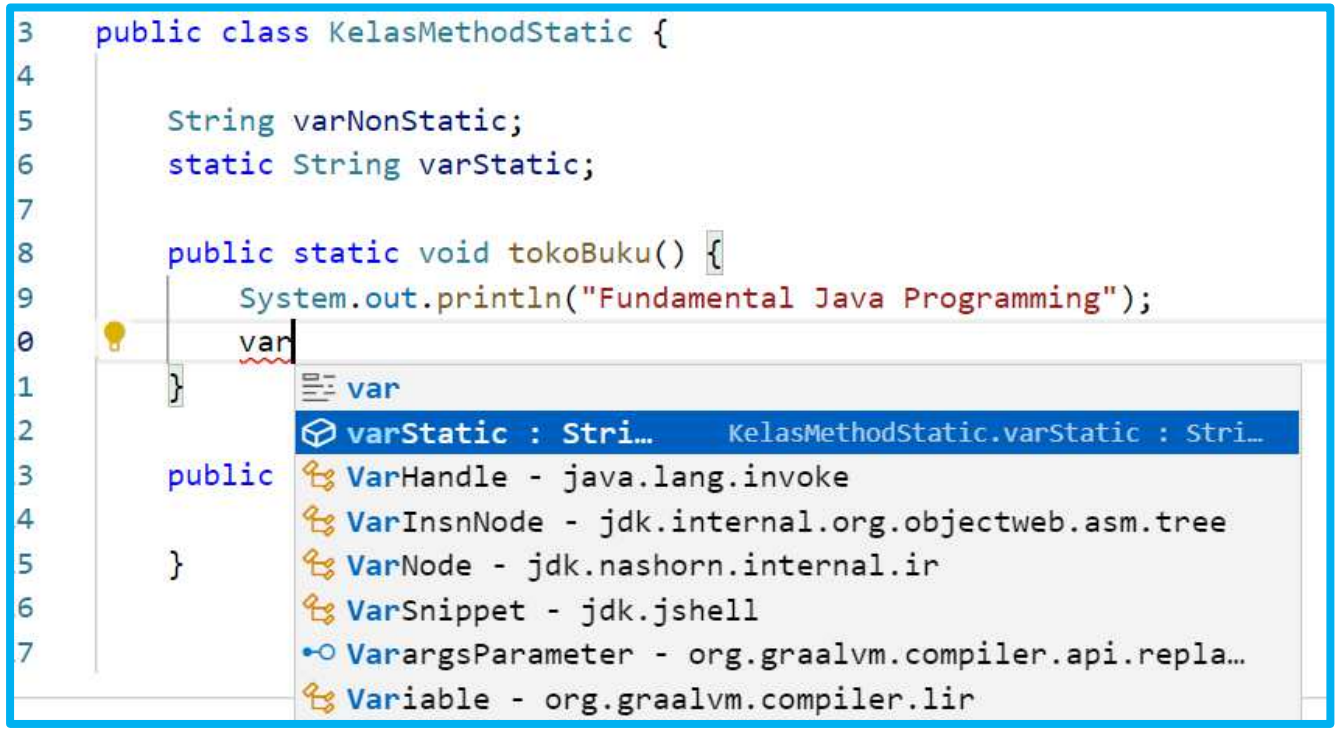
Method – Static

- Static Method adalah tipe method yang tidak memerlukan instance untuk pemanggilannya di method lain



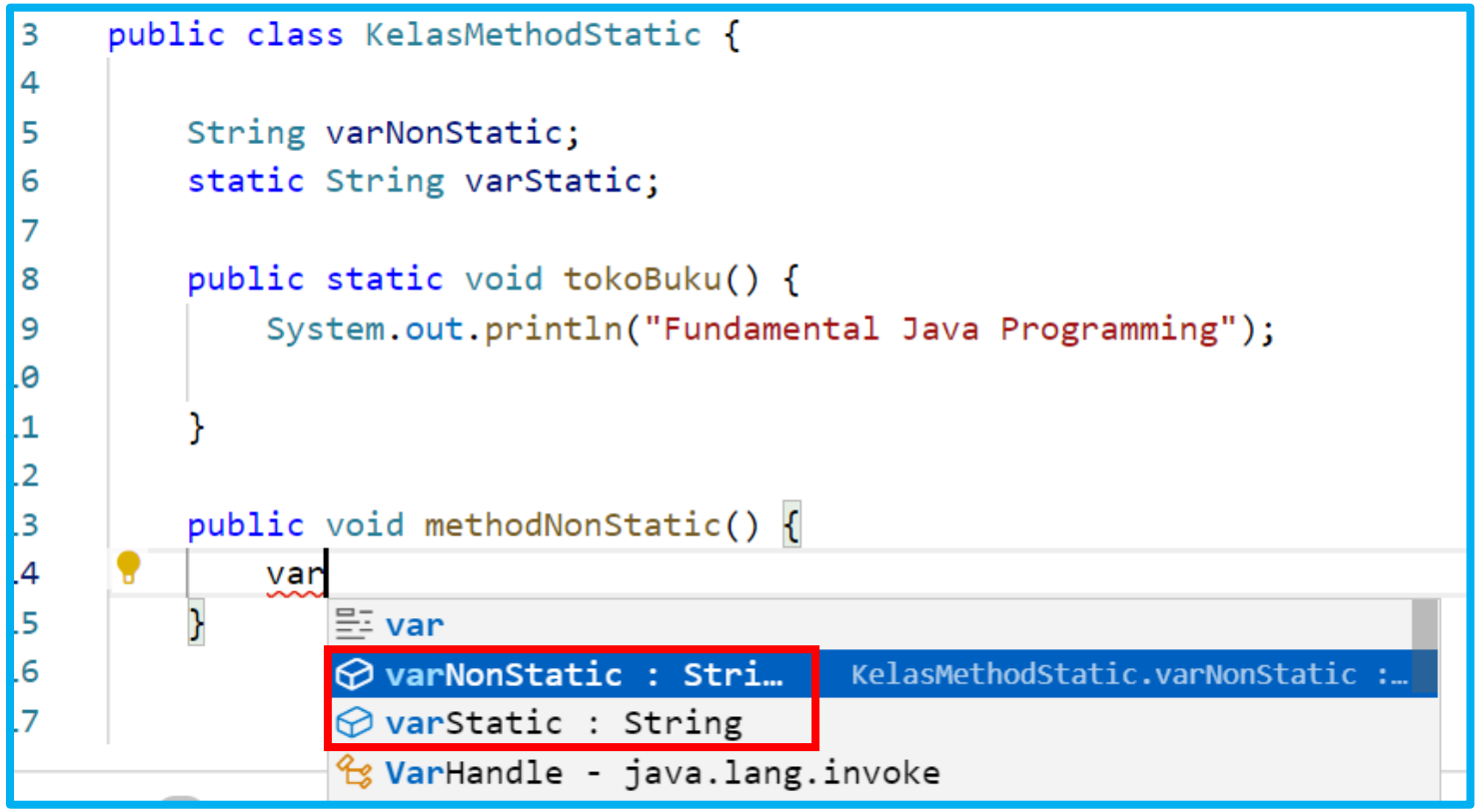
Method – Static – Access Variable

- Static Method tidak dapat mengakses property yang non-static, hanya static data member yang dapat diakses oleh static method.



Method – Non-Static – Access Variable

- Berbeda dengan non-static, dapat mengakses static data members.



Method – Function (Fungsi)

- Sebagai contoh, kita buah sebuah method di sebuah kelas.
- Method tsb dinamakan methodOutputString.
- Di dalam method tsb ada sebuah variable nama tipe String nilainya WILIAM.
- Var nama akan menjadi sebuah nilai output atau dilempar keluar ketika method itu dipanggil.

```
public void methodOutputString() {
    String nama = "WILIAM";

}
```

Method – Function (Fungsi)

- Lalu kita ketik return var nya di bagian akhir. Return nama berarti output nama.
- Lalu return nama akan error, ini karena methodnya masih void.

```
public void methodOutputString() {
    String nama = "WILIAM";
    return nama;
}
```

Method – Function (Fungsi)

- Lalu void kita ganti menjadi String. Mengapa? Karena return nama, var nama tipenya String.

```
public String methodOutputString() {
    String nama = "WILIAM";
    return nama;
}
```

Method – Function (Fungsi)

- Lalu kita buat method main

```
public class KelasMethodFunction {

    public String methodOutputString() {
        String nama = "WILIAM";
        return nama;
    }

    Run | Debug
    public static void main(String[] args) {
        // TODO Auto-generated method stub

    }

}
```

Method – Function (Fungsi)

- Lalu instance kelas tsb

```
public class KelasMethodFunction {

    public String methodOutputString() {
        String nama = "WILIAM";
        return nama;
    }

    Run | Debug
    public static void main(String[] args) {
        // TODO Auto-generated method stub
        KelasMethodFunction kmf = new KelasMethodFunction();

    }

}
```

Method – Function (Fungsi)

- Karena methodOutputString mempunyai return nama, maka kita buat var String nama untuk menampungnya.

```
1   package com.oop3;
2
3   public class KelasMethodFunction {
4
5       public String methodOutputString () {
6           String nama = "WILLIAM";
7           return nama;
8       }
9
        Run | Debug
10      public static void main(String[] args) {
11          // TODO Auto-generated method stub
12          KelasMethodFunction kmf = new KelasMethodFunction();
13          String nama = null;
14      }
15
16  }
```

Method – Function (Fungsi)

- Lalu var nama tersebut nilainya adalah methodOutputString.

```
1   package com.oop3;
2
3   public class KelasMethodFunction {
4
5       public String methodOutputString () {
6           String nama = "WILLIAM";
7           return nama;
8       }
9
        Run | Debug
10      public static void main(String[] args) {
11          // TODO Auto-generated method stub
12          KelasMethodFunction kmf = new KelasMethodFunction();
13          String nama = null;
14          nama = kmf.methodOutputString() ;
15      }
16
17  }
```

Method – Function (Fungsi)

- Lalu kita coba cetak.

```java
package com.oop3;

public class KelasMethodFunction {

    public String methodOutputString () {
        String nama = "WILLIAM";
        return nama;
    }

    Run | Debug
    public static void main(String[] args) {
        // TODO Auto-generated method stub
        KelasMethodFunction kmf = new KelasMethodFunction();
        String nama = null;
        nama = kmf.methodOutputString() ;
        System.out.println(nama);
    }

}
```

```
WILLIAM
```

Method – Function (Fungsi) - String

```java
package com.oop2;

public class KelasFunctionString {

    public String returnString() {
        String kota = "JAKARTA";
        return kota;
    }

    Run | Debug
    public static void main(String[] args) {
        // TODO Auto-generated method stub
        KelasFunctionString kfs = new KelasFunctionString();
        String city = null;
        city = kfs.returnString();
        System.out.println(city);
    }

}
```

```
JAKARTA
```

- Nama variable penampung tidak mesti selalu sama dengan nama variable return nya.
- Contoh nama var return adalah **kota**, sedangkan nama var penampung adalah **city**.

Method – Function (Fungsi) – String – Other Class

- Sama seperti inheritance, kelas lain dapat menggunakannya juga.

```java
package com.oop3;

import com.oop2.KelasFunctionString;

public class KelasFunctionStringBedaKelas {

    Run | Debug
    public static void main(String[] args) {
        // TODO Auto-generated method stub
        KelasFunctionString kfs = new KelasFunctionString() ;
        String kota = null;
        kota = kfs.returnString() + kfs.returnString() ;
        System.out.println(kota);
    }
}
```

```
JAKARTAJAKARTA
```

Method – Function (Fungsi) - int

```java
package com.oop3;

public class KelasFunctionInt {

    Run | Debug
    public static void main(String[] args) {
        // TODO Auto-generated method stub
        KelasFunctionInt kfi = new KelasFunctionInt() ;
        int angkaInt = 0;
        angkaInt = kfi.returnInt();
        System.out.println(angkaInt);
    }

    public int returnInt() {
        int angkaInt = 99;
        return angkaInt;
    }

}
```

```
99
```

Method – Function (Fungsi) - double

```java
package com.oop3;

public class KelasFunctionDouble {

    Run | Debug
    public static void main(String[] args) {
        // TODO Auto-generated method stub
        KelasFunctionDouble  kfi = new KelasFunctionDouble() ;
        double angkaDouble = 0;
        angkaDouble = kfi.returnDouble();
        System.out.println(angkaDouble);
    }

    public double returnDouble() {
        double angkaDouble = 99.999;
        return angkaDouble;
    }

}
```

```
99.999
```

Method – Function (Fungsi) - float

```java
package com.oop3;

public class KelasFunctionFloat {

    Run | Debug
    public static void main(String[] args) {
        // TODO Auto-generated method stub
        KelasFunctionFloat kfi = new KelasFunctionFloat();
        float angkaFloat = 0F;
        angkaFloat = kfi.returnFloat();
        System.out.println(angkaFloat);
    }

    public float returnFloat() {
        float angkaFloat = 99.99F;
        return angkaFloat;
    }

}
```

```
99.99
```

Method – Function (Fungsi) – More Than 1 Variable Return

- Bagaimana caranya ketika kita ingin return lebih dari satu variable dengan tipe yang sama?
- Kita bisa pakai array

Method – Function (Fungsi) – More Than 1 Variable Return

```java
package com.oop3;

public class KelasFunctionArray1DString {

    Run | Debug
    public static void main(String[] args) {
        // TODO Auto-generated method stub
        KelasFunctionArray1DString kfalds = new KelasFunctionArray1DString();
        String [] nama = new String[3];
        nama = kfalds.returnBanyakString();

        for (int i = 0; i < nama.length; i++) {
            System.out.println(nama[i]);
        }
    }
```

```
HARRY
TJOKROMULYO
WIYONO
```

Method – Function – Array 2D

```
1  package com.oop3;
2
3  public class KelasFunctionArray2DFloat {
4
       Run | Debug
5      public static void main(String[] args) {
6          // TODO Auto-generated method stub
7          KelasFunctionArray2DFloat kfa2df = new KelasFunctionArray2DFloat();
8          float[] [] banyakFloat = new float[3] [2];
9          banyakFloat = kfa2df.returnBanyakFloat();
10                 int baris - com.oop3.KelasFunctionArray2DFloat.main(String[])
11         for (int baris = 0; baris < 3; baris++) {
12             for (int kolom = 0; kolom < 2; kolom++) {
13                 System.out.print(banyakFloat[baris] [kolom]+" ");
14             }
15             System.out.println();
16         }
17     }
18
19     public float[] [] returnBanyakFloat() {
20         float[] [] banyakFloat = new float [3] [2];
21         banyakFloat[0] [0] = 11.11F; banyakFloat[0] [1] = 22.22F;
22         banyakFloat [1] [0] = 33.33F; banyakFloat [1] [1] = 44.44F;
23         banyakFloat [2] [0] = 55.55F; banyakFloat [2] [1] = 66.66F;
24
25         return banyakFloat;
26     }
27 }
```

```
11.11 22.22
33.33 44.44
55.55 66.66
```

Method – Function – ArrayList

```java
package com.oop3;

import java.util.ArrayList;
import java.util.List;

public class KelasFunctionArrayList {

    Run | Debug
    public static void main(String[] args) {
        // TODO Auto-generated method stub
        KelasFunctionArrayList kfal = new KelasFunctionArrayList();
        List<String> buahList = new ArrayList<String>();
        buahList = kfal.returnArrayList();

        for (int i = 0; i < buahList.size(); i++) {
            System.out.println(buahList.get(i));
        }
    }

    public List<String> returnArrayList() {
        List<String> buahList = new ArrayList<String>();
        buahList.add("SALAK");
        buahList.add("ANGGUR");
        buahList.add("PEPAYA");
        return buahList;
    }
}
```

```java
    public List<String> returnArrayList() {
        List<String> buahList = new ArrayList<String>();
        buahList.add("SALAK");
        buahList.add("ANGGUR");
        buahList.add("PEPAYA");
        return buahList;
    }
}
```

```
SALAK
ANGGUR
PEPAYA
```

Method – Function - Output

- Jadi simpulan sintax method dengan output adalah:

tipeAkses output namaMethod(){

// isi method

}

Contoh:

```
public void metodeSatu() {
    // tipeAkses : public
    // output : void (tanpa output)
    // namaMethod : metodeSatu
}
```

```
public String metodeDua() {
    // tipeAkses : public
    // output : String(halaman)
    // namaMethod : metodeDua
    String halaman = "DEPAN";
    return halaman;
}
```

Method – Function - Input

- Kita dapat memberikan suatu nilai input di suatu method
- Kita ingin kirim nilai dari var nama ke methodInput

```
1   package com.oop3;
2
3   public class KelasFunctionInputString {
4
        Run | Debug
5       public static void main(String[] args) {
6           // TODO Auto-generated method stub
7           KelasFunctionInputString kfis = new KelasFunctionInputString();
8           String nama = "CLARISSA";
9       }
10
11      public void methodInput(String nama) {
12
13      }
14
15  }
```

Method – Function - Input

- Kita ketik String nama di dalam ().
- Sekarang kita coba cetak var nama tapi di methodInput

```
1   package com.oop3;
2
3   public class KelasFunctionInputString {
4
        Run | Debug
5       public static void main(String[] args) {
6           // TODO Auto-generated method stub
7           KelasFunctionInputString kfis = new KelasFunctionInputString();
8           String nama = "CLARISSA";
9       }
10
11      public void methodInput() {
12
13      }
14
15  }
```

Method – Function - Input

- Kita taruh sintax cetak di methodInput

```
1   package com.oop3;
2
3   public class KelasFunctionInputString {
4
        Run | Debug
5       public static void main(String[] args) {
6           // TODO Auto-generated method stub
7           KelasFunctionInputString kfis = new KelasFunctionInputString();
8           String nama = "CLARISSA";
9       }
10
11      public void methodInput(String nama) {
12          System.out.println(nama);
13      }
14
15  }
```

Method – Function - Input

- Lalu kita panggil methodInput, maka muncul methodInput(String nama)

```
1   package com.oop3;
2
3   public class KelasFunctionInputString {
4
        Run | Debug
5       public static void main(String[] args) {
6           // TODO Auto-generated method stub
7           KelasFunctionInputString kfis = new KelasFunctionInputString();
8           String nama = "CLARISSA";
9       kfis.method
10          }
11
12      public void
13          System.
14      }
15
16  }
```

```
methodInput(String nama) …   KelasFunctionInputString…
public_method                Public method
private_method               private_method
protected_method             Protected Method
public_static_method         Public static method
private_static_method        Private static method
```

Method – Function - Input

```java
package com.oop3;

public class KelasFunctionInputString {

    Run | Debug
    public static void main(String[] args) {
        // TODO Auto-generated method stub
        KelasFunctionInputString kfis = new KelasFunctionInputString();
        String nama = "CLARISSA";

        kfis.methodInput(nama);
    }

    public void methodInput(String nama) {
        System.out.println(nama);
    }

}
```

```
CLARISSA
```

Method – Function – Input – Many Var – Same Type

```java
package com.oop3;

public class KelasFunctionInputManySameTypeVar {

    Run | Debug
    public static void main(String[] args) {
        // TODO Auto-generated method stub
        KelasFunctionInputManySameTypeVar kfimstv = new KelasFunctionInputManySameTypeVar();
        String namaDepan = "RAMLAN";
        String namaBelakang = "NASUHA";

        kfimstv.methodInput(namaDepan, namaBelakang);
    }

    public void methodInput(String namaDepan, String namaBelakang) {
        String namaLengkap = namaDepan + namaBelakang;
        System.out.println(namaLengkap);
    }

}
```

```
RAMLANNASUHA
```

Method – Function – Input – Many Var – Other Type

```java
package com.oop3;

public class KelasFunctionInputManyOtherTypeVar {

    Run | Debug
    public static void main(String[] args) {
        // TODO Auto-generated method stub
        KelasFunctionInputManyOtherTypeVar kfimotv = new KelasFunctionInputManyOtherTypeVar();
        String nama = "STEPHANIE";
        int tinggi = 164;
        double ipk = 3.42;
        kfimotv.methodInput(nama, tinggi, ipk);
    }

    public void methodInput(String nama, int tinggi, double ipk) {
        System.out.println(nama);
        System.out.println(tinggi);
        System.out.println(ipk);
    }

}
```

```
STEPHANIE
164
3.42
```

Method – Function – Output Input

- Jadi simpulan sintax method dengan input adalah:

tipeAkses output namaMethod(input){

// isi method

}

Contoh:

```
public void metodeSatu() {
    // tipeAkses : public
    // output : void
    // namaMethod : metodeSatu
    // input () : tanpa Input
}
```

```
public void metodeDua(String nama, int usia) {
    // tipeAkses : public
    // output : void
    // namaMethod : metodeSatu
    // input () : String nama dan int usia

}
```

Method – Function – Output Input

KelasFunctionInputOutput.java

```
1  package dasar.oop3;
2
3  public class KelasFunctionInputOutput {
4
5      public static void main(String[] args) {
6          // TODO Auto-generated method stub
7
8          KelasFunctionInputOutput kfio = new KelasFunctionInputOutput();
9          int nilai1 = 108;
10         int nilai2 = 202;
11         int hasil = kfio.tambah(nilai1, nilai2);
12         System.out.println(hasil);
13
14     }
15
16     public int tambah(int nilai1, int nilai2) {
17         int hasil = nilai1 + nilai2;
18         return hasil;
19     }
20
21 }
```

Console

<terminated> Kelas

```
310
```

Polymorphism

- Banyak Bentuk
- Antara Superclass dengan Subclass memiliki nama method yang sama
- Terbagi dua:
    1. Overriding
    2. Overloading

Polymorphism - Overriding

- Nama method di superclass dan subclass sama
- Fungsi / content dari method dari superclass diubah di subclass

Polymorphism – Overriding

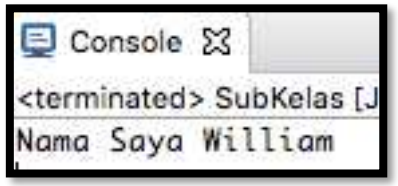
SuperKelas.java

```
package oop;

public class SuperKelas {

    private void cetakNama() {
        System.out.println("Nama Saya William");
    }

}
```

SubKelas.java

```
package oop;

public class SubKelas extends SuperKelas {

    public void cetakNama() {
        System.out.println("Nama Saya Wijaya");

    }

    public static void main(String[] args) {
        // TODO Auto-generated method stub
        SuperKelas sk = new SuperKelas();
        sk.cetakNama();
    }

}
```

Output



Polymorphism – Overriding

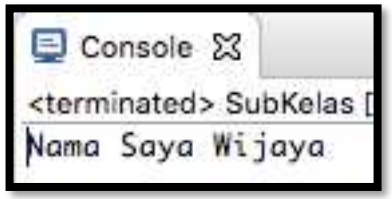
SuperKelas.java

```java
package oop;

public class SuperKelas {

    private void cetakNama() {
        System.out.println("Nama Saya William");
    }

}
```

SubKelas.java

```java
package oop;

public class SubKelas extends SuperKelas {

    public void cetakNama() {
        System.out.println("Nama Saya Wijaya");

    }

    public static void main(String[] args) {
        // TODO Auto-generated method stub
        //SuperKelas sk = new SuperKelas();
        //sk.cetakNama();

        // Overriding
        SuperKelas sk = new SubKelas();
        sk.cetakNama();
    }

}
```

Output



Polymorphism - Overloading

- Nama method dengan method lainnya sama
- Parameter dari setiap methodnya berbeda.

Polymorphism - Overloading

SuperKelas.java

```java
package oop;

public class OverloadingKelas {

    static void hitung(int a, int b) {
        int c = a + b;
        System.out.println(c);
    }

     static void hitung(String a, String b) {
        String c = a + b;
        System.out.println(c);
    }

    public static void main(String[] args) {
        // TODO Auto-generated method stub

        //parameter int
        hitung(2,4);

        //parameter String
        hitung("2","2");

    }

}
```