

# Polymorphism dan Interface

1. Polimorfisme dalam OOP adalah sebuah prinsip di mana class dapat memiliki banyak “Bentuk” method yang berbeda-beda meskipun namanya sama. “Bentuk” di artikan: isinya berbeda, parameternya berbeda, dan tipe datanya berbeda. Fungsi utamanya polimorfisme yaitu dapat memiliki method yang berbeda beda dalam satu class meskipun method nya memiliki nama yang sama namun isinya didalam method yang namanya sama itu memiliki nilai variabel yang berbeda beda. Polimorfisme pada Java ada dua macam:
  - a. Static Polymorphism (Polimorfisme statis)  
Polimorfisme statis menggunakan method overloading yang parameternya berbeda dan biasanya dalam satu class.
  - b. Dynamic Polymorphism (Polimorfisme dinamis).  
polimorfisme dinamis menggunakan method overriding yang isinya/body method berbeda dan biasanya beda class.
2. Interface adalah class yang tidak memiliki tubuh pada method-methodnya. Method interface tersebut harus diimplementasikan dalam kelas turunannya tidak boleh tidak. Dalam pengimplementasiannya bisa saja tiap-tiap class akan mengimplementasikan secara berbeda dengan nama method yang sama. Di dalam interface, deklarasi variable memiliki atribut final sehingga bersifat absolut. Keyword final inilah yang menjadi keunikan sendiri bagi interface bahwa ouput dari bagian interface berupa final yang tidak diganti pada saat implementasi kecuali di override. interface berfungsi sebagai penghubung antara sesuatu yang ‘abstrak’ dengan sesuatu yang nyata.

## Program Listing:

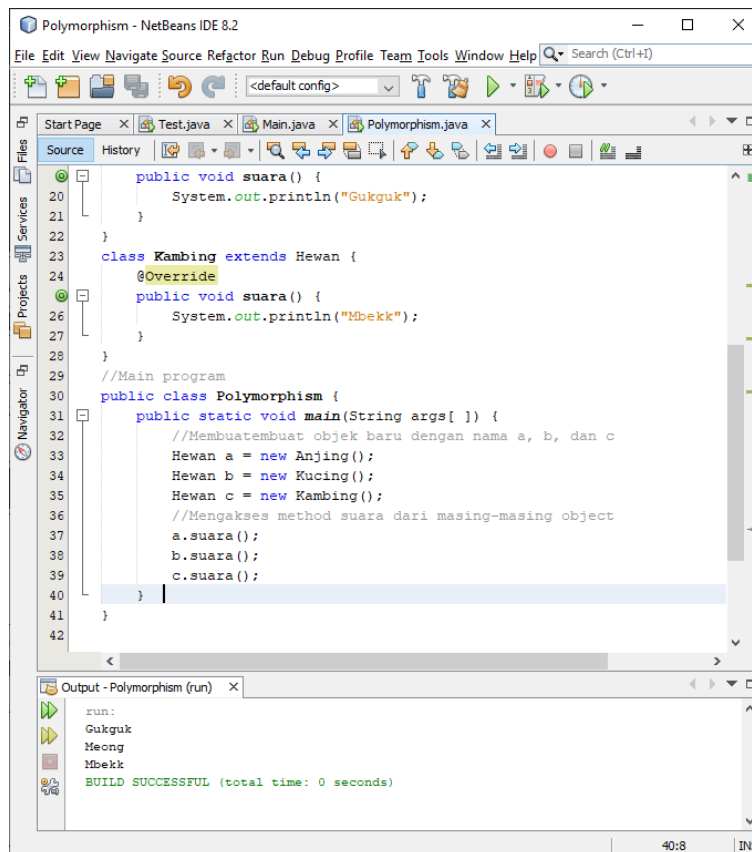
### 1. Polimorfisme dinamis.

```
package polymorphism;
// Membuat class induk dengan nama hewan
class Hewan {
    //Membuat method void suara dengan modifier public
    public void suara() {
        System.out.println("....");
    }
}
//Membuat class kucing yang merupakan turunan dari class hewan
class Kucing extends Hewan {
    //Membuat method overriding
    @Override
    public void suara() {
        System.out.println("Meong");
    }
}
class Anjing extends Hewan {
    @Override
    public void suara() {
        System.out.println("Gukguk");
    }
}
class Kambing extends Hewan {
    @Override
    public void suara() {
        System.out.println("Mbekk");
    }
}
//Main program
public class Polymorphism {
    public static void main(String args[ ]) {
        //Membuat objek baru dengan nama a, b, dan c
        Hewan a = new Anjing();
        Hewan b = new Kucing();
        Hewan c = new Kambing();
        //Mengakses method suara dari masing-masing object
        a.suara();
    }
}
```

```

        b.suara();
        c.suara();
    }
}

```



Pada program dan gambar diatas merupakan penerapan dari konsep polimorfisme dinamis yang membuat kelas turunan dari induk class hewan yang memiliki method suara. Pada class turunan kemudian mendeklarasikan ulang method suara dari class induk dengan isi bunyi yang berbeda-beda pada setiap class hewan turunan yang bisa disebut method overriding atau yang bisa disebut juga polimorfisme dinamis.

## 2. Polimorfisme statis.

```

package polymorphism;
//Membuat class induk tampilan
class tampilan{
    //membuat method void suara dengan modifier public
    public void print(int angka){
        System.out.println("ini adalah angka :" + angka);
    }
    //membuat method yang sama tetapi berbeda parameter

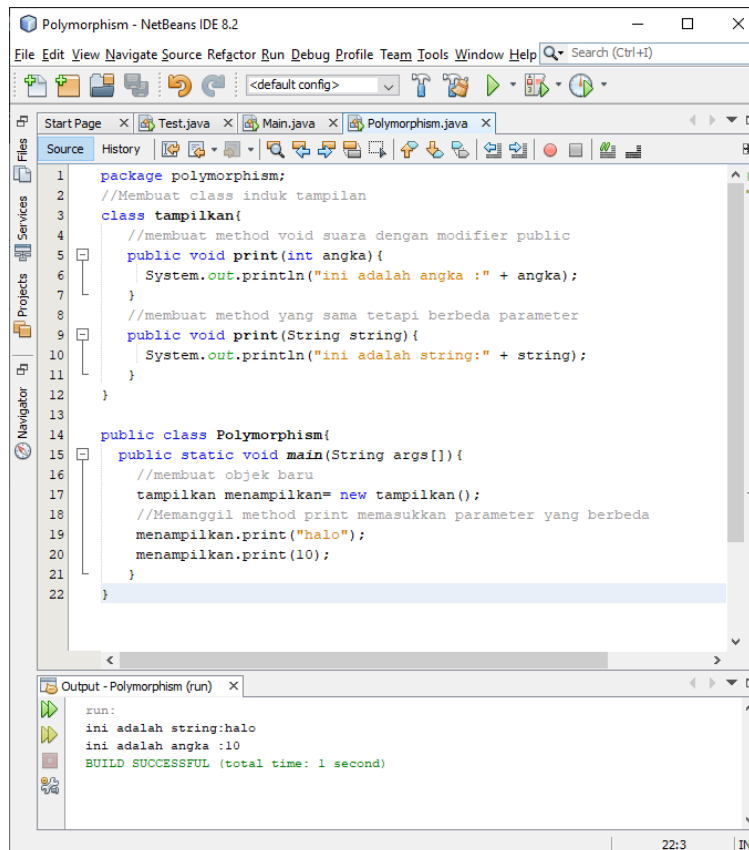
```

```

    public void print(String string){
        System.out.println("ini adalah string:" + string);
    }
}

public class Polymorphism{
    public static void main(String args[]){
        //membuat objek baru
        tampilan menampilkan= new tampilan();
        //Memanggil method print memasukkan parameter yang berbeda
        menampilkan.print("halo");
        menampilkan.print(10);
    }
}

```



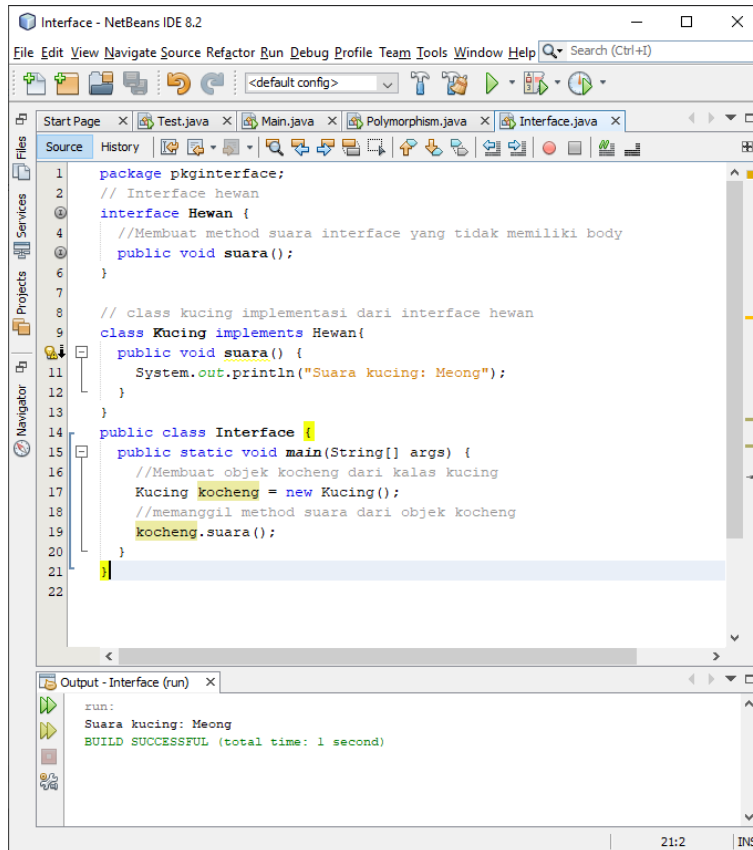
Pada program kali ini merupakan penerapan dari konsep polimorfisme statis. Pada program tersebut membuat class tampilan yang memiliki method dengan nama yang sama yaitu print tetapi memiliki parameter yang berbeda ini yang bisa disebut method overloading atau yang bisa disebut juga polimorfisme statis.

### 3. Interface

```
package pkginterface;
// Interface hewan
interface Hewan {
    //Membuat method suara interface yang tidak memiliki body
    public void suara();
}

// class kucing implementasi dari interface hewan
class Kucing implements Hewan{
    public void suara() {
        System.out.println("Suara kucing: Meong");
    }
}

public class Interface {
    public static void main(String[] args) {
        //Membuat objek kocheng dari kelas kucing
        Kucing kocheng = new Kucing();
        //memanggil method suara dari objek kocheng
        kocheng.suara();
    }
}
```



```
1 package pkginterface;
2 // Interface hewan
3 interface Hewan {
4     //Membuat method suara interface yang tidak memiliki body
5     public void suara();
6 }
7
8 // class kucing implementasi dari interface hewan
9 class Kucing implements Hewan{
10     public void suara() {
11         System.out.println("Suara kucing: Meong");
12     }
13 }
14
15 public class Interface {
16     public static void main(String[] args) {
17         //Membuat objek kocheng dari kelas kucing
18         Kucing kocheng = new Kucing();
19         //memanggil method suara dari objek kocheng
20         kocheng.suara();
21     }
22 }
```

Output - Interface (run)

```
run:
Suara kucing: Meong
BUILD SUCCESSFUL (total time: 1 second)
```

Pada program diatas merupakan penerapan konsep interface. Interface hewan tersebut memilike method interface yang tidak memiliki body. Untuk menggunakan interface hewan harus "diimplementasikan" terlebih dahulu dengan kata kunci implements. Isi metode interface disediakan oleh class kucing yang telah diimplementasikan.