

# PRAKTIKUM 11

## INHERITANCE 2



### A. TUJUAN PEMBELAJARAN

1. Melakukan pengontrolan akses pada pengkodean
2. Menggunakan kata kunci `super`
3. Menghindari kesalahan pada pewarisan konstruktor

### B. DASAR TEORI

Suatu parent class dapat tidak mewariskan sebagian member-nya kepada subclass-nya. Sejauh mana suatu member dapat diwariskan ke class lain, ataupun suatu member dapat diakses dari class lain, sangat berhubungan dengan access control (kontrol pengaksesan). Di dalam java, kontrol pengaksesan dapat digambarkan dalam Tabel 9.1.

Tabel 9.1. Akses modifier

Modifier	class yang sama	package yang sama	subclass	class manapun
private	√			
default	√ 	√		
protected	√ 	√	√	
public	√	√	√	√

Kata kunci *super* dipakai untuk merujuk pada member dari parent class, sebagaimana kata kunci *this* yang dipakai untuk merujuk pada member dari class itu sendiri. Adapun format penulisannya adalah sebagai berikut:


`super.data_member` → merujuk pada data member pada parent class

`super.function_member()` → merujuk pada function member pada parent class



super()

→ merujuk pada konstruktor pada parent class

Dalam inheritance, member kelas yang diwariskan hanyalah **variabel** dan **method** saja. Sedangkan **konstruktorparent** class tidak diwariskan ke sub class. Tetapi ketika suatu obyek anak dibuat dalam artian ketika konstruktor anak dijalankan maka konstruktor parent class dijalankan terlebih dahulu dan selanjutnya menyelesaikan konstruktor anak. 





### C. TUGAS PENDAHULUAN

1. Ada berapa modifier untuk pengontrolan akses? Jelaskan masing-masing!
2. Apakah kegunaan kata kunci super? Jelaskan !
3. Apakah yang dimaksud dengan konstruktor tidak diwariskan?

### D. PERCOBAAN

#### Percobaan 1 : Menggunakan kata kunci super

Berikut ini listing penggunaan kata kunci super.

```
class Parent {  
    public int x = 5;  
}  
  
class Child extends Parent {  
    public int x = 10;  
    public void Info(int x) {  
        System.out.println("Nilai x sebagai parameter = " + x)   
        System.out.println("Data member x di class Child = " + this.x);   
        System.out.println("Data member x di class Parent = " + super.x);   
    }  
}  
  
public class NilaiX {  
    public static void main(String args[]) {  
        Child tes = new Child();   
        tes.Info(20);  
    }  
}
```

```
}
```

Ketika program tersebut dijalankan, akan tampak hasil seperti dibawah ini :

```
Nilai x sebagai parameter = 20
Data member x di class Child = 10
Data member x di class Parent = 5
```

## Percobaan 2 : Konstruktor tidak diwariskan

Buatlah class kosong bernama Parent seperti dibawah:

```
public class Parent {


}
```

Buatlah class Child yang menurunkan class Parent seperti dibawah ini:

```
public class Child extends Parent {
    int x;
    public Child() {
        x = 5;
        super();
    }
}
```

Lakukan kompilasi pada Child diatas. Apa yang terjadi?. Pasti disana terjadi error.

Sekarang ubahlah sedikit class Child diatas seperti dibawah ini:

```
public class Child extends Parent {
    int x;
    public Child() {
        super(); 
        x = 5;
    }
}
```

Setelah dikompilasi, anda tidak mendapatkan error sebagaimana yang sebelumnya. Ini yang harus kita perhatikan bahwa untuk pemanggilan konstruktor parent class, kita harus melakukan pemanggilan tersebut di baris pertama pada konstruktor subclass.

## E. LATIHAN

### Latihan 1. Konstruktor tidak diwariskan

Buatlah class berikut ini

```
class Base{
    Base(int i){
        System.out.println("base constructor");
    }
    Base(){
    }
}
```

```
public class Sup extends Base{
    public static void main(String argv[]){
        Sup s= new Sup();
        //baris 1
    }

    Sup(){
        // baris 2
    }

    public void derived(){
        // baris 3
    }
}
```

Modifikasilah class Sup (di bagian **//baris 1**, **//baris 2** atau **//baris 3**) sedemikian hingga konstruktor kelas **Base** (konstruktor **Base(int i)**) dipanggil dan menampilkan string “base constructor” ke layar!

## Latihan 2. Konstruktor tidak diwariskan

```
private class Base{  
    Base(){  
        int i = 100;  
        System.out.println(i);  
    }  
}
```

```
public class Pri extends Base{  
    static int i = 200;  
    public static void main(String argv[]){  
        Pri p = new Pri();  
        System.out.println(i);  
    }  
}
```

Kompilasi dan jalankan program diatas! Apa yang terjadi? Jelaskan !

## Latihan 3. Apa yang tampil di layar bila kode dibawah ini dijalankan?

```
class X{  
    Y b = new Y();  
    X(){  
        System.out.print("X");  
    }  
}
```

```
class Y{  
    Y(){  
        System.out.print("Y");  
    }  
}
```

```
public class Z extends X{  
    Y y = new Y();  
    Z(){  
        System.out.print("Z");  
    }  
    public static void main(String[] args){  
        new Z();  
    }  
}
```

#### Latihan 4. Kompile dan jalankan program berikut! Apa yang terjadi? Jelaskan !

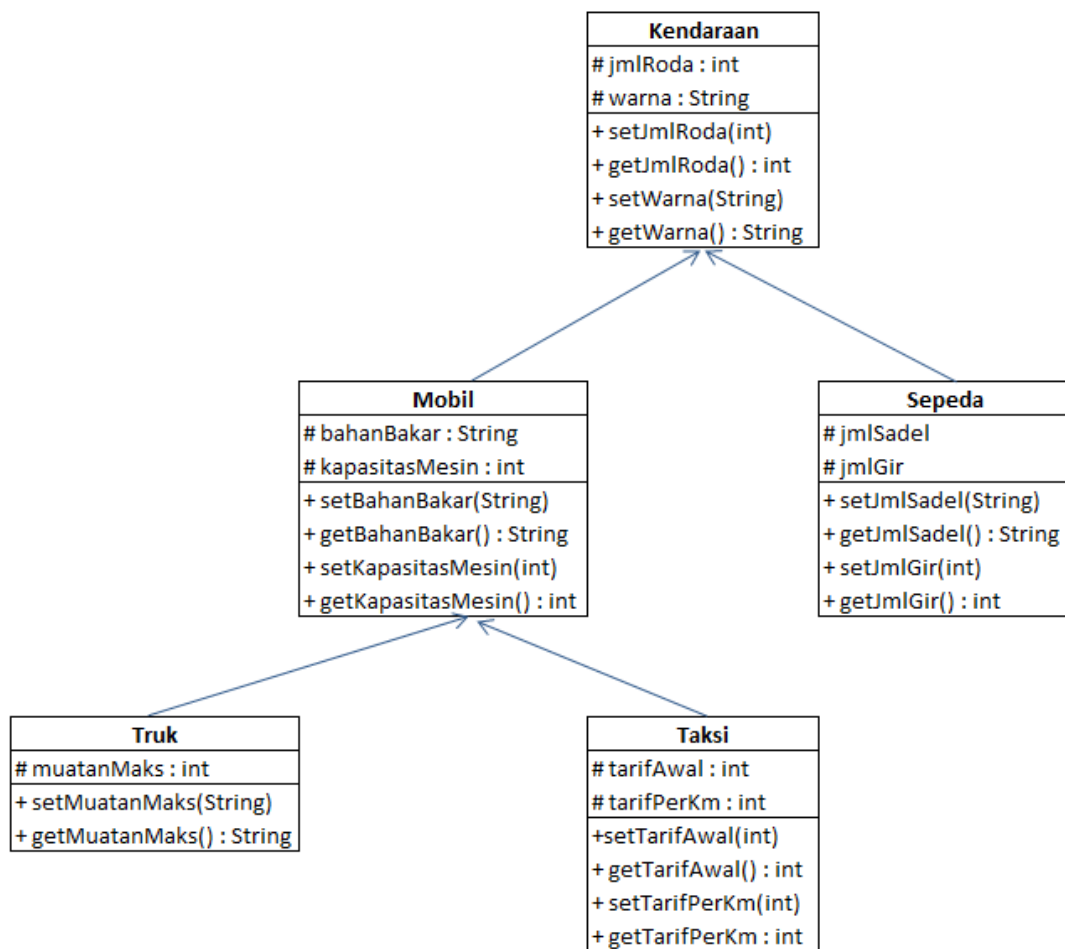
```
public class Hope{
    public static void main(String argv[]){
        Hope h = new Hope();
    }

    protected Hope(){
        for(int i =0; i <10; i ++){
            System.out.println(i);
        }
    }
}
```

#### F. TUGAS

Mengimplementasikan UML class diagram dalam program.

a. Buatlah kelas-kelas berdasarkan UML class diagram dibawah ini!



- b. Selanjutnya buatlah kelas Tes.java yang membuat obyek-obyek serta mengeset nilai variabel seperti pada Tabel 9.2.dan tampilkan data per obyek.

Tabel 9.2. Data obyek

Obyek	jmlRoda	warna	bahanBakar	kapasitasMesin	muatanMaks	
truk1	4	kuning	solar	1500	1000	
truk2	6	merah	solar	2000	5000	
					tarifAwal	tarifPerKm
taksi1	4	oranye	bensin	1500	10000	5000
taksi1	4	biru	bensin	1300	7000	3500
			jmlSadel	jmlGir		
sepeda1	3	hitam	1	2		
sepeda2	2	putih	2	5		

## G. LAPORAN RESMI

Kumpulkan hasil latihan dan tugas di atas. Tambahkan analisa dalam laporan resmi.