

# JOBSHEET W06

## INHERITANCE

### 1. KOMPETENSI

1. Memahami konsep dasar inheritance atau pewarisan.
2. Mampu membuat suatu subclass dari suatu superclass tertentu.
3. Mampu mengimplementasikan konsep hierarchical inheritance
4. Mampu membuat objek dari suatu subclass dan melakukan pengaksesan terhadap atribut dan method baik yang dimiliki sendiri atau turunan dari superclass nya.

### 2. PENDAHULUAN

**Inheritance** pada object oriented programming merupakan konsep **pewarisan** dari suatu class yang lebih umum ke suatu class yang lebih spesifik. Kelas yang menurunkan disebut kelas dasar (**base class/super class/parent class**), sedangkan kelas yang diturunkan disebut kelas turunan (**derived class/sub class/child class**). Setiap **subclass** akan “mewarisi” atribut dan method dari **superclass** yang bersifat *public* ataupun *protected*. Manfaat pewarisan adalah *reusability* atau penggunaan kembali baris kode.

Pada bahasa pemrograman Java, deklarasi inheritance dilakukan dengan cara menambahkan kata kunci **extends** setelah deklarasi nama class, kemudian diikuti dengan nama parent class-nya. Kata kunci **extends** tersebut memberitahu kompiler Java bahwa kita ingin melakukan **extension/ perluasan** class. Berikut adalah contoh deklarasi inheritance.

```
public class B extends A {  
    ...  
}
```

Contoh diatas memberitahukan kompiler Java bahwa class B meng-**extend** class A. Artinya, class B adalah subclass dari class A dengan melakukan extension/perluasan. Extension atau perluasan ini akan dilakukan dengan penambahan atribut dan method khusus yang hanya dimiliki oleh class B.

Suatu parent class bisa membatasi atribut dan method yang akan diwariskan kepada subclass-nya. Pembatasan tersebut dilakukan melalui penentuan access level modifier. Di dalam java, access level modifier atribut dan method dirangkum dalam tabel berikut ini:

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

Atribut dan method yang akan diwariskan dari parent class ke child class adalah atribut dan method dengan modifier *protected* atau *public*.

Kata kata kunci **this** digunakan untuk merujuk pada current object/class. Sementara kata kunci **super** digunakan untuk merujuk pada parent object/class. Format penulisannya adalah sebagai berikut:

- **super.<namaAtribut>**  
Mengakses atribut parent
- **super.<namaMethod>()**  
Memanggil method parent

## 1. PERCOBAAN 1 (extends)

### A. TAHAPAN PERCOBAAN

1. Buatlah sebuah parent class dengan nama Pegawai. Lalu buat constructor tanpa parameter dengan baris kode sebagai berikut:

```
public class Pegawai {

    public Pegawai() {
        System.out.println("Objek dari class Pegawai dibuat");
    }

}
```

2. Buatlah subclass dari class Pegawai dengan nama Dosen, kemudian buat juga constructor tanpa parameter dengan baris kode berikut:

```
public class Dosen extends Pegawai {

    public Dosen() {
        System.out.println("Objek dari class Dosen dibuat");
    }

}
```

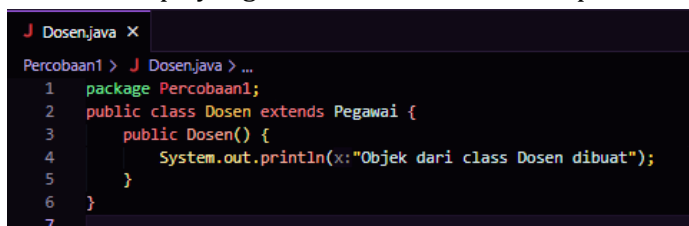
3. Buatlah main class, misal InheritanceDemo.java, lakukan instansiasi objek baru bernama dosen1 dari class Dosen sebagai berikut:

```
public static void main(String[] args) {
    Dosen dosen1 = new Dosen();
}
```

4. Run programnya kemudian amati hasilnya.

### B. PERTANYAAN

1. Pada percobaan 1 diatas, tentukan child class dan parent class!  
**The Lecturer and Employee classes are two separate classes that do not show any direct inheritance relationship.**
2. Kata kunci apa yang membuat child class dan parent class tersebut memiliki relasi?



```
J Dosen.java x
Percobaan1 > J Dosen.java > ...
1 package Percobaan1;
2 public class Dosen extends Pegawai {
3     public Dosen() {
4         System.out.println("Objek dari class Dosen dibuat");
5     }
6 }
7
```

**extends keyword is required. If Lecturer is a child class of Employee**

3. Berdasarkan hasil yang ditampilkan oleh program, ada berapa constructor yang dieksekusi? Constructor class mana yang lebih dulu dieksekusi?  
**because adding inheritance using the extends keyword then there will be two constructors executed:**

**The constructor of the Employee class will be executed first (because this class is**

the parent class).

After that, the constructor of the Dosen class (child class) will be executed

```
de\User\workspaceStorage\b8bc508368f2360eb
Percobaan1.InheritanceDemo'
Objek dari class Pegawai dibuat
Objek dari class Dosen dibuat
PS C:\Users\HP\Downloads\jobsheet6>
```

## 4. PERCOBAAN 2 (Pewarisan)

### A. TAHAPAN PERCOBAAN

1. Tambahkan atribut nip, nama, dan gaji serta method getInfo() pada class Pegawai

```
public class Pegawai {
    public String nip;
    public String nama;
    public double gaji;

    public Pegawai() {
        System.out.println("Objek dari class Pegawai dibuat");
    }

    public String getInfo(){
        String info = "";
        info += "NIP          : " + nip + "\n";
        info += "Nama          : " + nama + "\n";
        info += "Gaji           : " + gaji + "\n";

        return info;
    }
}
```



```
1 package Percobaan2;
2
3 public class Pegawai {
4     public String nip;
5     public String nama;
6     public double gaji;
7
8     public Pegawai () {
9         System.out.println ("Objek dari class Pegawai dibuat");
10    }
11    public String getInfo () {
12        String info = "";
13        info += "NIP      : " + nip + "\n";
14        info += "Nama      : " + nama + "\n";
15        info += "Gaji      : " + gaji + "\n";
16
17        return info;
18    }
19
20 }
21
```

2. Tambahkan pula atribut NIDN pada class Dosen

```
public class Dosen extends Pegawai {
    public String nidn;

    public Dosen() {
        System.out.println("Objek dari class Dosen dibuat");
    }
}
```

3. Pada class InheritanceDemo.java tuliskan baris kode berikut:

```
public static void main(String[] args) {
    Dosen dosen1 = new Dosen();

    dosen1.nama = "Yansy Ayuningtyas";
    dosen1.nip = "34329837";
    dosen1.gaji = 3000000;
    dosen1.nidn = "1989432439";

    System.out.println(dosen1.getInfo());
}
```



```
1 package Percobaan2;
2
3 public class InheritanceDemo {
4     public static void main(String[] args) {
5         Dosen dosen1 = new Dosen ();
6
7         dosen1.nama = "Yansy Ayuningtyas";
8         dosen1.nip = "34329837";
9         dosen1.gaji = 3000000;
10        dosen1.nidn = "1989432439";
11
12        System.out.println(dosen1.getInfo());
13    }
14 }
15
```

4. Run program kemudian amati hasilnya

## B. PERTANYAAN

1. Pada percobaan 2 diatas, apakah program dapat berhasil dijalankan ataukah terjadi error?

**the program will run successfully and no errors will occur.**

2. Jika program berhasil dijalankan, mengapa tidak terjadi error pada assignment/pengisian nilai atribut nip, gaji, dan NIDN pada object dosen1 padahal tidak ada deklarasi ketiga atribut tersebut pada class Dosen?

**The nip, name, and salary attributes can be accessed by the dosen1 object because the modifier is public, and the nidn attribute is declared directly in the Dosen class.**

3. Jika program berhasil dijalankan, mengapa tidak terjadi error pada pemanggilan method getInfo() oleh object dosen1 padahal tidak ada deklarasi method getInfo() pada class Dosen?

**There is no error when calling the getInfo() method by the lecturer1 object because getInfo() can be called by the lecturer1 object because the Lecturer class inherits this method from the Employee class, and the method is public.**

```
aan2.InheritanceDemo'  
Objek dari class Pegawai dibuat  
Objek dari class Dosen dibuat  
NIP    : 34329837  
Nama   : Yansy Ayuningtyas  
Gaji   : 3000000.0  
  
PS C:\Users\HP\Downloads\jobsheet6>
```

## 5. PERCOBAAN 3 (Hak akses)

### A. TAHAPAN PERCOBAAN

1. Modifikasi access level modifier pada atribut gaji menjadi private pada class Pegawai.java

```
public class Pegawai {  
    public String nip;  
    public String nama;  
    private double gaji;  
}
```

2. Run program kemudian amati hasilnya.
3. Ubah access level modifier atribut gaji menjadi protected kemudian pindah class Pegawai ke package baru, misalnya "testpackage".

```
package testpackage;  
  
public class Pegawai {  
    public String nip;  
    public String nama;  
    protected double gaji;  
}
```

4. Import class Pegawai dari testpackage pada class Dosen.

```
package inheritance;  
import testpackage.Pegawai;
```

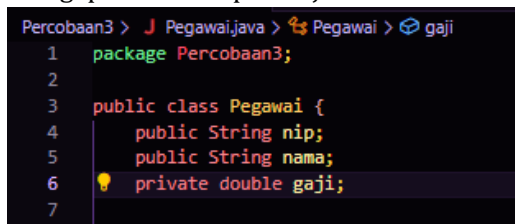
5. Akses atribut gaji pada class Dosen dengan coba mencetak atribut gaji pada constructor Dosen

```
public Dosen() {  
    System.out.println(gaji);  
    System.out.println("Objek dari class Dosen dibuat");  
}
```

6. Ubah kembali access level modifier menjadi public dan kembalikan class Pegawai ke package semula.

### B. PERTANYAAN

1. Pada langkah 1 di atas, terjadi error karena object dosen1 tidak dapat mengakses atribut gaji. Padahal gaji merupakan atribut Pegawai yang merupakan parent class dari Dosen. Mengapa hal ini dapat terjadi?



**because the salary attribute in the Employee class is declared with the private modifier. private can only be accessed in the class where the attribute is declared**

2. Pada langkah 5, setelah class Pegawai berpindah ke package yang berbeda, class Dosen masih dapat mengakses atribut gaji. Mengapa?

**The Dosen class can still access the salary attribute after the Employee class is moved to a different package because the salary is changed to protected. The protected modifier allows access by the class itself, subclasses, and other classes in the same package. Thus, attribute inheritance still works even though it is in a different package.**

3. Berdasarkan percobaan tersebut, bagaimana menentukan atribut dan method yang akan diwariskan oleh parent class ke child class?

**The attributes and methods inherited by a child class are determined by its access modifiers:**

**Private is not inherited.**

**Protected allows access in subclasses, even in different packages.**

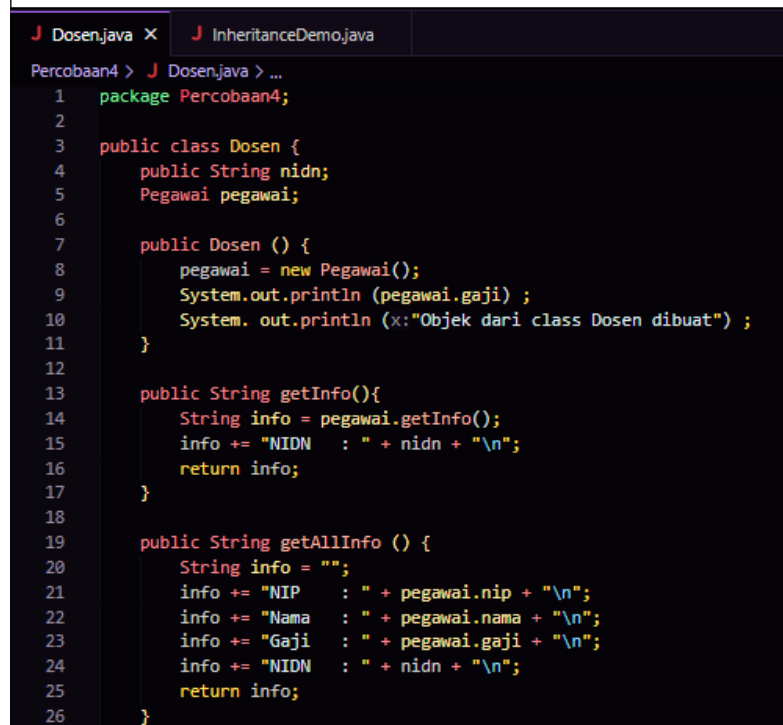
**Public is inherited and can be accessed from anywhere.**

## 6. PERCOBAAN 4 (Super - atribut)

### A. TAHAPAN PERCOBAAN

1. Butlah method getAllInfo() pada class Dosen

```
public String getAllInfo() {  
    String info = "";  
    info += "NIP      : " + nip + "\n";  
    info += "Nama      : " + nama + "\n";  
    info += "Gaji      : " + gaji + "\n";  
    info += "NIDN      : " + nidn + "\n";  
  
    return info;  
}
```



The screenshot shows an IDE with two tabs: 'Dosen.java' and 'InheritanceDemo.java'. The 'Dosen.java' tab is active, showing the following code:

```
Percobaan4 > J Dosen.java > ...  
1  package Percobaan4;  
2  
3  public class Dosen {  
4      public String nidn;  
5      Pegawai pegawai;  
6  
7      public Dosen () {  
8          pegawai = new Pegawai();  
9          System.out.println (pegawai.gaji) ;  
10         System. out.println (x:"Objek dari class Dosen dibuat") ;  
11     }  
12  
13     public String getInfo(){  
14         String info = pegawai.getInfo();  
15         info += "NIDN  : " + nidn + "\n";  
16         return info;  
17     }  
18  
19     public String getAllInfo () {  
20         String info = "";  
21         info += "NIP      : " + pegawai.nip + "\n";  
22         info += "Nama      : " + pegawai.nama + "\n";  
23         info += "Gaji      : " + pegawai.gaji + "\n";  
24         info += "NIDN      : " + nidn + "\n";  
25         return info;  
26     }  
}
```

2. Lakukan pemanggilan method getAllInfo() oleh object dosen1 pada class InheritanceDemo.java

```
public static void main(String[] args) {  
    Dosen dosen1 = new Dosen();  
  
    dosen1.nama = "Yansy Ayuningtyas";  
    dosen1.nip = "34329837";  
    dosen1.gaji = 3000000;  
    dosen1.nidn = "1989432439";  
  
    System.out.println(dosen1.getAllInfo());  
}
```



```
J Dosen.java  J InheritanceDemo.java x
Percobaan4 > J InheritanceDemo.java > InheritanceDemo > main(String[])
1  package Percobaan4;
2
3  public class InheritanceDemo {
4      Run | Debug
5      public static void main (String[] args) {
6          Dosen dosen1 = new Dosen () ;
7          dosen1.pegawai.nama = "Yansy Ayuningtyas";
8          dosen1.pegawai.nip = "34329837";
9          dosen1.pegawai.gaji = 3000000;
10         dosen1.nidn = "1989432439";
11
12         System.out.println (dosen1.getAllInfo () ) ;
13     }
14 }
15
```

- Run program kemudian amati hasilnya

```
obaan4.InheritanceDemo'
Objek dari class Pegawai dibuat
0.0
Objek dari class Dosen dibuat
NIP   : 34329837
Nama  : Yansy Ayuningtyas
Gaji  : 3000000.0
NIDN  : 1989432439
PS C:\Users\HP\Downloads\jobsheet6>
```

- Lakukan modifikasi method `getAllInfo()` pada class `Dosen`

```
public String getAllInfo() {
    String info = "";
    info += "NIP       : " + this.nip + "\n";
    info += "Nama      : " + this.nama + "\n";
    info += "Gaji       : " + this.gaji + "\n";
    info += "NIDN      : " + this.nidn + "\n";

    return info;
}
```

- Run program kemudian bandingkan hasilnya dengan langkah no 2.

```
heet6_591b83a8\bin' 'Percobaan4.InheritanceDemo'
Objek dari class Pegawai dibuat
0.0
Objek dari class Dosen dibuat
NIP   : 34329837
Nama  : Yansy Ayuningtyas
Gaji  : 3000000.0
NIDN  : 1989432439
PS C:\Users\HP\Downloads\jobsheet6>
```

- Lakukan modifikasi method `getAllInfo()` pada class `Dosen` kembali

```

public String getAllInfo() {
    String info = "";
    info += "NIP      : " + super.nip + "\n";
    info += "Nama      : " + super.nama + "\n";
    info += "Gaji      : " + super.gaji + "\n";
    info += "NIDN      : " + super.nidn + "\n";

    return info;
}

```

```

public String getAllInfo() {
    String info = "";
    info += "NIP      : " + super.nip + "\n";
    info += "Nama      : " + super.nama + "\n";
    info += "Gaji      : " + super.gaji + "\n";
    info += "NIDN      : " + super.nidn + "\n";

    return info;
}

```

7. Run program kemudian bandingkan hasilnya dengan program pada no 1 dan no 4.

```

obaan4.InheritanceDemo
Objek dari class Pegawai dibuat
Objek dari class Dosen dibuat
Exception in thread "main" java.lang.Error: Unresolved compilation problem:
    nidn cannot be resolved or is not a field

    at Percobaan4.Dosen.getAllInfo(Dosen.java:55)
    at Percobaan4.InheritanceDemo.main(InheritanceDemo.java:12)
PS C:\Users\HP\Downloads\jobsheet6>

```

8. Lakukan modifikasi method getAllInfo() pada class Dosen kembali

```

public String getAllInfo() {
    String info = "";
    info += "NIP      : " + super.nip + "\n";
    info += "Nama      : " + super.nama + "\n";
    info += "Gaji      : " + super.gaji + "\n";
    info += "NIDN      : " + this.nidn + "\n";

    return info;
}

```

```

public String getAllInfo() {
    String info = "";
    info += "NIP      : " + super.nip + "\n";
    info += "Nama      : " + super.nama + "\n";
    info += "Gaji      : " + super.gaji + "\n";
    info += "NIDN      : " + this.nidn + "\n";

    return info;
}

```

- Run program kemudian bandingkan hasilnya dengan program pada no 2 dan no 4.

**program 1 and 4 have the same results, all information such as NIP, Name, Salary, and NIDN are displayed correctly. While in program 8, initially there were no values for NIP, Name, Salary, and NIDN displayed.**

```
obaan4.InheritanceDemo'  
Objek dari class Pegawai dibuat  
Objek dari class Dosen dibuat  
NIP : 34329837  
Nama : Yansy Ayuningtyas  
Gaji : 3000000.0  
NIDN : 1989432439  
  
PS C:\Users\HP\Downloads\jobsheet6>
```

## B. PERTANYAAN

- Apakah terdapat perbedaan hasil nama, nip, dan gaji yang ditampilkan pada program 1, 4, dan 8? Mengapa?

**There is a Difference: The results displayed in programs 1 and 4 are the same, while program 8 experiences an error in displaying the initial value.**

- Mengapa error terjadi pada program no 6?

**Incorrect use of super and this**

## 7. PERCOBAAN 5 (super & overriding)

### A. TAHAPAN PERCOBAAN

- Lakukan modifikasi kembali pada method getAllInfo(). Run program kemudian amati hasilnya

```
public String getAllInfo() {  
    String info = getInfo();  
    info += "NIDN : " + nidn;  
  
    return info;  
}
```

```
Percobaan5 > J Dosen.java > Dosen > getAllInfo()  
1 package Percobaan5;  
2  
3 public class Dosen extends Pegawai {  
4     public String nidn;  
5     Pegawai pegawai;  
6  
7     public Dosen () {  
8         pegawai = new Pegawai();  
9         System.out.println (pegawai.gaji) ;  
10        System.out.println (x:"Objek dari class Dosen dibuat") ;  
11    }  
12    public String getAllInfo() {  
13        String info = getInfo();  
14        info += "NIDN : " + nidn + "\n";  
15        return info;  
16    }  
17 }  
18
```

```
obaan5.InheritanceDemo'  
Objek dari class Pegawai dibuat  
Objek dari class Pegawai dibuat  
0.0  
Objek dari class Dosen dibuat  
NIP : null  
Nama : null  
Gaji : 0.0  
NIDN : 1989432439  
  
PS C:\Users\HP\Downloads\jobsheet6>
```

2. Lakukan modifikasi kembali pada method `getAllInfo()`. Run program kemudian amati hasilnya

```
public String getAllInfo() {  
    String info = this.getInfo();  
    info += "NIDN      : " + nidn;  
  
    return info;  
}
```

```
public String getAllInfo() {  
    String info = this.getInfo();  
    info += "NIDN : " + nidn + "\n";  
    return info;  
}
```

```
4ecbe6\redhat.java\jdt_ws\jobsheet6_591b83a  
Objek dari class Pegawai dibuat  
Objek dari class Pegawai dibuat  
0.0  
Objek dari class Dosen dibuat  
NIP      : null  
Nama     : null  
Gaji     : 0.0  
NIDN     : 1989432439  
  
PS C:\Users\HP\Downloads\jobsheet6>
```

3. Lakukan modifikasi kembali pada method `getAllInfo()`. Run program kemudian amati hasilnya

```
public String getAllInfo() {  
    String info = super.getInfo();  
    info += "NIDN      : " + nidn;  
  
    return info;  
}
```

```
public String getAllInfo () {  
    String info = super.getInfo();  
    info += "NIDN : " + nidn + "\n";  
    return info;  
}
```

```
4ecbe6\redhat.java\jdt_ws\jobsheet6_591b83a8V  
Objek dari class Pegawai dibuat  
Objek dari class Pegawai dibuat  
0.0  
Objek dari class Dosen dibuat  
NIP      : 34329837  
Nama     : Yansy Ayuningtyas  
Gaji     : 3000000.0  
NIDN     : 1989432439  
  
PS C:\Users\HP\Downloads\jobsheet6> █
```

4. Tambahkan method `getInfo()` pada class `Dosen` dan modifikasi method `getAllInfo()` sebagai berikut

```
public class Dosen extends Pegawai {  
    public String nidn;  
  
    public Dosen() {  
        System.out.println("Objek dari class Dosen dibuat");  
    }  
  
    public String getInfo(){  
        return "NIDN      : " + this.nidn + "\n";  
    }  
  
    public String getAllInfo(){  
        String info = super.getInfo();  
        info += this.getInfo();  
  
        return info;  
    }  
}
```

```

34 //modify no4
35 public class Dosen extends Pegawai {
36     public String nidn;
37
38     public Dosen() {
39         System.out.println(x:"Objek dari class Dosen dibuat");
40     }
41
42     public String getInfo() {
43         return "NIDN : " + this.nidn + "\n";
44     }
45
46     public String getAllInfo() {
47         String info = super.getInfo();
48         info += this.getInfo();
49         return info;
50     }
51 }
52

```

```

4ecbe6\redhat.java\jdt_ws\jobsheet6_591b83a8\bin'
Objek dari class Pegawai dibuat
Objek dari class Dosen dibuat
NIP   : 34329837
Nama  : Yansy Ayuningtyas
Gaji  : 3000000.0
NIDN  : 1989432439

PS C:\Users\HP\Downloads\jobsheet6>

```

## B. PERTANYAAN

- Apakah ada perbedaan method `getInfo()` yang diakses pada langkah 1, 2, dan 3?  
**Step 1 and 2: Calling `getInfo()` from Lecturer without setting the attribute, so the result is the default value.**  
**Step 3: Calling `super.getInfo()` but because the Lecturer attribute value is already set, the correct information is displayed.**
- Apakah ada perbedaan method `super.getInfo()` dan `this.getInfo()` yang dipanggil dalam method `getAllInfo()` pada langkah 4? Jelaskan!  
**The main difference is where the methods are called from. `super.getInfo()` accesses a method from the parent class, while `this.getInfo()` accesses a method from the current class (Dosen), which means it returns more specific information.**
- Pada method apakah terjadi overriding? Jelaskan!  
**Overriding: Is a process in which a subclass (in this case, Lecturer) provides a new implementation for a method that has been defined in the superclass (in this case, Employee).**  
**Overriding occurs in the `getInfo()` method in the Dosen class.**

## 8. PERCOBAAN 6 (overloading)

### A. TAHAPAN PERCOBAAN

- Tambahkan constructor baru untuk class Dosen sebagai berikut

```

public Dosen(String nip, String nama, double gaji, String nidn){
    System.out.print("Objek dari class Dosen dibuat dengan constructor berparameter");
}

```

```

Percobaan6 > J Dosen.java > Dosen > getAllInfo()
3 public class Dosen {
4     public String nIdn;
5     Pegawai pegawai;
6
7     public Dosen (String nip, String nama, double gaji, String nIdn) {
8         pegawai = new Pegawai(nip,nama,gaji);
9         this.nIdn=nIdn;
10        System.out.println("Objek dari class Dosen dibuat dengan constructor berparameter") ;
11    }
12
13    public Dosen () {
14        pegawai = new Pegawai();
15        System.out.println (pegawai.gaji) ;
16        System. out.println (x:"Objek dari class Dosen dibuat") ;
17    }
18
19    public String getInfo(){
20        // String info = pegawai.getInfo();
21        // info += "NIDN    : " + nIdn + "\n";
22        // return info;
23        return "NIDN    : " + nIdn + "\n";
24    }
25
26    public String getAllInfo () {
27        String info = pegawai.getInfo();
28        info += this.getInfo();
29        return info;
30    }

```

2. Modifikasi class InheritanceDemo untuk menginstansiasi object baru dengan nama dosen2 dengan constructor yang berparameter. Run program kemudian amati hasilnya.

The parameterized constructor in the Dosen class is used to directly set the values of NIP, Name, Salary, and NIDN when the object is created.

Create a dosen2 object with a parameterized constructor and display complete lecturer information using the getAllInfo() method.

```

public static void main(String[] args) {
    Dosen dosen2 = new Dosen ("34329837", "Yansy Ayuningtyas", 3000000, "1989432439");
    System.out.println(dosen2.getAllInfo());
}

```

```

J InheritanceDemo.java x
Percobaan6 > J InheritanceDemo.java > {} Percobaan6
1 package Percobaan6;
2
3 public class InheritanceDemo {
4     Run | Debug
5     public static void main (String[] args) {
6         Dosen dosen2 = new Dosen (nip:"34329837", nama:"Yansy Ayuningtyas", gaji:3000000, nIdn:"1989432439") ;
7         System. out.println (dosen2.getAllInfo()) ;
8     }

```

```

PS C:\Users\HP\Downloads\jobsheet6> c:; cd 'c:\Users\HP\Downloads\jobsheet6
Objek dari class pegawai dibuat dengan constructor berparameter
Objek dari class Dosen dibuat dengan constructor berparameter
NIP    : 34329837
Nama    : Yansy Ayuningtyas
Gaji    : 3000000.0
NIDN    : 1989432439

PS C:\Users\HP\Downloads\jobsheet6>

```

## B. PERTANYAAN

1. Bagaimana hasil nilai nip, nama, gaji, dan nidn yang ditampilkan pada langkah 2? Mengapa demikian?

**The NIP, Name, Salary, and NIDN values will be displayed according to the parameters given when creating the dosen2 object. This is because the constructor with parameters directly initializes these values when the object is created.**

2. Jelaskan apakah constructor tanpa parameter dan constructor class Dosen yang dibuat pada langkah 1 memiliki signature yang sama?

**The parameterless constructor and the constructor created in step 1 do not have the same signature. The signatures are different because the parameterless constructor does not accept arguments, while the constructor in step 1 accepts four parameters (nip, name, salary, and nidn).**

3. Konsep apa dalam OOP yang membolehkan suatu class memiliki constructor atau method dengan nama yang sama dan signature yang berbeda pada satu class?

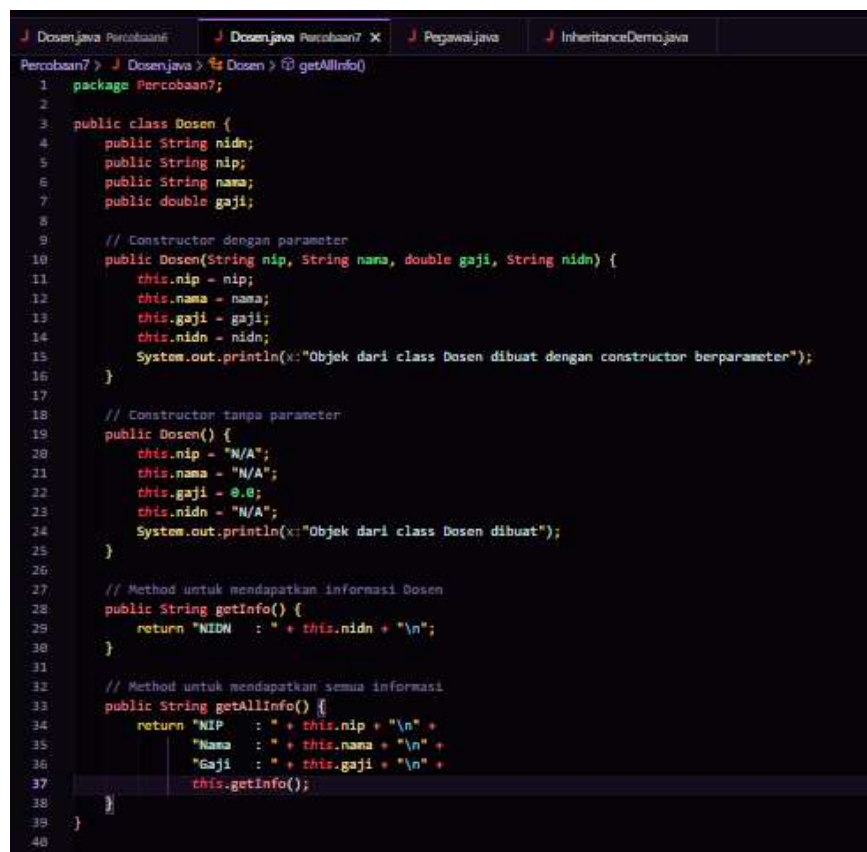
Overloading allows a class to have multiple methods or constructors with the same name, but with different signatures (number or type of parameters).

## 9. PERCOBAAN 7 (super - constructor)

### A. TAHAPAN PERCOBAAN

1. Modifikasi constructor pada class Dosen sebagai berikut. Run program kemudian amati hasilnya.

```
public Dosen(String nip, String nama, double gaji, String nidn){
    this.nip = nip;
    this.nama = nama;
    this.gaji = gaji;
    this.nidn = nidn;
}
```



```
Dosen.java Percobaan7
Dosen.java Percobaan7 x Pegawai.java InheritanceDemo.java
Percobaan7 > J Dosen.java > F Dosen > gettAllInfo()
1 package Percobaan7;
2
3 public class Dosen {
4     public String nidn;
5     public String nip;
6     public String nama;
7     public double gaji;
8
9     // Constructor dengan parameter
10    public Dosen(String nip, String nama, double gaji, String nidn) {
11        this.nip = nip;
12        this.nama = nama;
13        this.gaji = gaji;
14        this.nidn = nidn;
15        System.out.println("Objek dari class Dosen dibuat dengan constructor berparameter");
16    }
17
18    // Constructor tanpa parameter
19    public Dosen() {
20        this.nip = "N/A";
21        this.nama = "N/A";
22        this.gaji = 0.0;
23        this.nidn = "N/A";
24        System.out.println("Objek dari class Dosen dibuat");
25    }
26
27    // Method untuk mendapatkan informasi Dosen
28    public String getInfo() {
29        return "NIDN : " + this.nidn + "\n";
30    }
31
32    // Method untuk mendapatkan semua informasi
33    public String gettAllInfo() {
34        return "NIP : " + this.nip + "\n" +
35            "Nama : " + this.nama + "\n" +
36            "Gaji : " + this.gaji + "\n" +
37            this.getInfo();
38    }
39 }
40
```



```

obaan7.InheritanceDemo'
Objek dari class Dosen dibuat dengan constructor berparameter
NIP : 34329837
Nama : Yansy Ayuningtyas
Gaji : 3000000.0
NIDN : 1989432439

```

```
PS C:\Users\HP\Downloads\jobsheet6>
```

2. Modifikasi constructor pada class Dosen sebagai berikut. Run program kemudian amati hasilnya.

```

public Dosen(String nip, String nama, double gaji, String nidn){
    super.nip = nip;
    super.nama = nama;
    super.gaji = gaji;
    this.nidn = nidn;
}

```

```

// Modify 2
public class Dosen extends Pegawai {
    public String nidn;

    public Dosen(String nip, String nama, double gaji, String nidn) {
        super(nip, nama, gaji);
        this.nidn = nidn;
        System.out.println(x:"Objek dari class Dosen dibuat dengan constructor berparameter"
    }

    // Method untuk mendapatkan informasi Dosen
    public String getInfo() {
        return "NIDN : " + this.nidn + "\n";
    }

    // Method untuk mendapatkan semua informasi
    public String getAllInfo() {
        return super.getInfo() + this.getInfo();
    }
}

```

```

anceDemo'
Objek dari class Dosen dibuat dengan constructor
NIP : 34329837
Nama : Yansy Ayuningtyas
Gaji : 3000000.0
NIDN : 1989432439

```

```
PS C:\Users\HP\Downloads\jobsheet6>
```

3. Modifikasi constructor pada class Dosen sebagai berikut. Run program kemudian amati hasilnya.

```

public Dosen(String nip, String nama, double gaji, String nidn){
    super();
    super.nip = nip;
    super.nama = nama;
    super.gaji = gaji;
    this.nidn = nidn;
}

```

```
// Modify 3

public class Dosen extends Pegawai {
    public String nidn;

    public Dosen(String nip, String nama, double gaji, String nidn) {
        super();
        super.nip = nip;
        super.nama = nama;
        super.gaji = gaji;
        this.nidn = nidn;
        System.out.println(x:"Objek dari class Dosen dibuat dengan constructor berparameter"
    }

    // Method untuk mendapatkan informasi Dosen
    public String getInfo() {
        return "NIDN : " + this.nidn + "\n";
    }

    public String getAllInfo() {
        return super.getInfo() + this.getInfo();
    }
}
```

```
InheritanceDemo
Exception in thread "main" java.lang.Error: Unresolved compilation problem:
The constructor Pegawai() is undefined

    at Percobaan7.Dosen.<init>(Dosen.java:71)
    at Percobaan7.InheritanceDemo.main(InheritanceDemo.java:5)
PS C:\Users\HP\Downloads\jobsheet6> []
```

4. Hapus/comment constructor tanpa parameter dari class Pegawai. Tambahkan constructor baru untuk class Pegawai sebagai berikut. Run program kemudian amati hasilnya.

```
public class Pegawai {
    public String nip;
    public String nama;
    public double gaji;

    // public Pegawai() {
    //     System.out.println("Objek dari class Pegawai dibuat");
    // }

    public Pegawai(String nip, String nama, double gaji) {
        this.nip = nip;
        this.nama = nama;
        this.gaji = gaji;
    }

    public String getInfo(){
        String info = "";
        info += "NIP : " + nip + "\n";
        info += "Nama : " + nama + "\n";
        info += "Gaji : " + gaji + "\n";

        return info;
    }
}
```

```

Dosen.java Percobaan6  Dosen.java Percobaan7  Pegawai.java X  InheritanceDemo.java
Percobaan7 > J Pegawai.java > Pegawai
1  package Percobaan7;
2
3  public class Pegawai {
4      public String nip;
5      public String nama;
6      public double gaji;
7
8
9      public Pegawai(String nip, String nama, double gaji) {
10         this.nip = nip;
11         this.nama = nama;
12         this.gaji = gaji;
13     }
14     public String getInfo() {
15         String info = "";
16         info += "NIP : " + nip + "\n";
17         info += "Nama : " + nama + "\n";
18         info += "Gaji : " + gaji + "\n";
19         return info;
20     }
21 }

```

```

anceDemo'
Objek dari class Dosen dibuat dengan constructor berparameter
NIP : 34329837
Nama : Yansy Ayuningtyas
Gaji : 3000000.0
NIDN : 1989432439

PS C:\Users\HP\Downloads\jobsheet6>
Launchpad 0 1 W 0 Java: Ready

```

- Modifikasi constructor pada class Dosen sebagai berikut. Run program kemudian amati hasilnya.

```

public Dosen(String nip, String nama, double gaji, String nidn){
    this.nidn = nidn;
    super(nip, nama, gaji);
}

```

```

//Modify 5 dan 6

public class Dosen extends Pegawai {
    public String nidn;

    public Dosen(String nip, String nama, double gaji, String nidn) {
        this.nidn = nidn;
        super(nip, nama, gaji);
        System.out.println("Objek dari class Dosen dibuat dengan constructor berparameter");
    }

    // Method untuk mendapatkan informasi Dosen
    @Override
    public String getInfo() {
        return super.getInfo() + "NIDN : " + this.nidn + "\n";
    }

    // Method untuk mendapatkan semua informasi
    public String getAllInfo() {
        return this.getInfo();
    }
}

```

```

C:\Users\HP\Downloads\jobsheet6\redhat.java\jdt_ws\jobsheet6_591b83a8\bin' 'Percobaan7.InheritanceDemo'
Exception in thread "main" java.lang.Error: Unresolved compilation problems:
Implicit super constructor Pegawai() is undefined. Must explicitly invoke another constructor
Constructor call must be the first statement in a constructor

at Percobaan7.Dosen.<init>(Dosen.java:93)
at Percobaan7.InheritanceDemo.main(InheritanceDemo.java:5)
PS C:\Users\HP\Downloads\jobsheet6>

```

6. Modifikasi constructor pada class Dosen sebagai berikut. Run program kemudian amati hasilnya.

```
public Dosen(String nip, String nama, double gaji, String nidn){
    super(nip, nama, gaji);
    this.nidn = nidn;
}
```

```
public class Dosen extends Pegawai {
    public String nidn;

    public Dosen(String nip, String nama, double gaji, String nidn) {
        super(nip, nama, gaji);
        this.nidn = nidn;
        System.out.println("Objek dari class Dosen dibuat dengan constructor berparameter");
    }

    // Method untuk mendapatkan informasi Dosen
    @Override
    public String getInfo() {
        return super.getInfo() + "NIDN : " + this.nidn + "\n";
    }

    // Method untuk mendapatkan semua informasi
    public String getAllInfo() {
        return this.getInfo();
    }
}
```

```
spaceStorage\b8bc508368f2360eb4ccc102f84ecbe6\redhat.java\jdt_ws\jobsheet6\anceDemo
Objek dari class Dosen dibuat dengan constructor berparameter
NIP : 34329837
Nama : Yansy Ayuningtyas
Gaji : 3000000.0
NIDN : 1989432439

PS C:\Users\HP\Downloads\jobsheet6>
Launchpad 0 1 0 Java: Ready
```

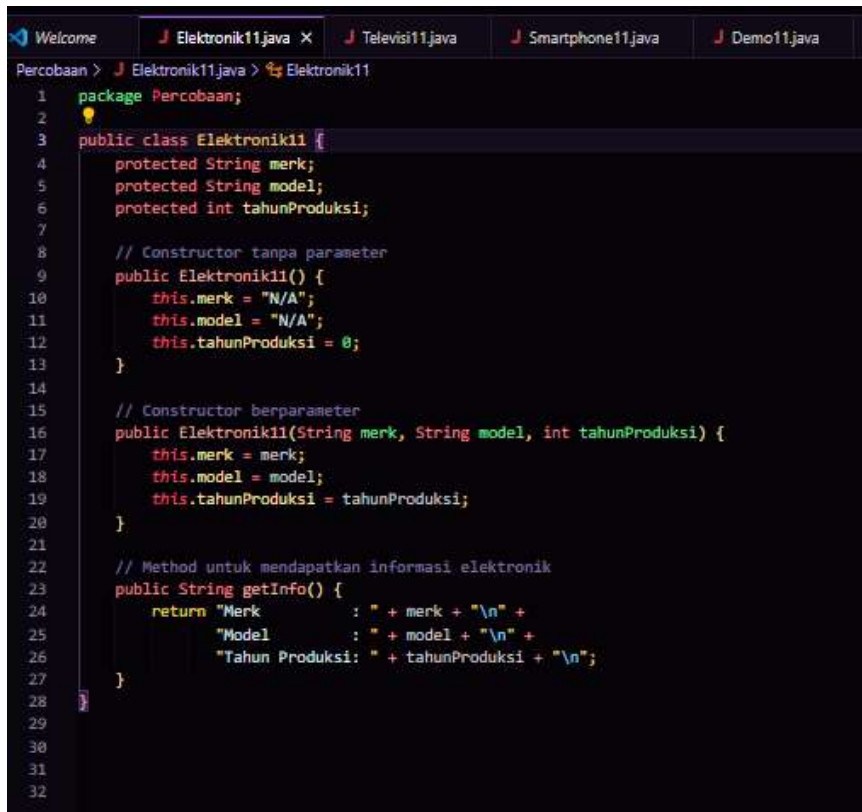
## B. PERTANYAAN

1. Apakah terdapat perbedaan hasil pada langkah 1 dan 2? Jelaskan!  
**Tidak ada, hasil Ketika di tampilkan sama**
2. Apakah terdapat perbedaan hasil pada langkah 2 dan 3? Jelaskan!  
**Ya ada, perbedaan terjadi Ketika Langkah 2 bisa menampilkan hasil, sedangkan Langkah 3 mengalami error karena The constructor Pegawai() is undefined**
3. Mengapa terjadi error pada langkah 4?  
**The error in step 4 occurs because you are calling the parameterless constructor of the Employee class implicitly with super(), but that constructor is not defined.**
4. Apa perbedaan super() yang dipanggil pada langkah 3 dan 6?  
**Constructor Call: In step 3, super() without parameters is invalid if the default constructor is missing from the parent class. In contrast, in step 6, super(nip, nama, gaji) calls the appropriate constructor.**  
**Proper Initialization: Step 6 ensures that all parent class attributes are properly initialized, whereas step 3 does not do so effectively.**
5. Mengapa terjadi error pada langkah 5?  
**this.nidn = nidn; is now set first to initialize the nidn attribute.**  
**super(nip, nama, gaji); is called afterward to call the constructor of the Employee class and initialize the nip, nama, and gaji attributes.**

## 10. TUGAS

1. Tentukan sebuah class yang merupakan turunan dari class yang lain.
2. Buat 3 atribut pada parent class kemudian tambahkan minimal 1 atribut pada child class.
3. Lakukan method overloading dengan membuat 2 constructor yaitu constructor tanpa parameter dan constructor berparameter pada masing-masing class. Panggil constructor `super()` berparameter untuk membuat object dari parent class pada constructor child class.
4. Implementasikan class diagram yang dibuat pada mata kuliah PBO teori
5. Buat class Demo kemudian lakukan instansiasi objek child class pada main function
6. Cobalah melakukan modifikasi nilai atribut (baik yang dideklarasikan pada child class maupun yang diwariskan dari kemudian print info nya).

### a. Kelas Elektronik



```
1 package Percobaan;
2
3 public class Elektronik11 {
4     protected String merk;
5     protected String model;
6     protected int tahunProduksi;
7
8     // Constructor tanpa parameter
9     public Elektronik11() {
10         this.merk = "N/A";
11         this.model = "N/A";
12         this.tahunProduksi = 0;
13     }
14
15     // Constructor berparameter
16     public Elektronik11(String merk, String model, int tahunProduksi) {
17         this.merk = merk;
18         this.model = model;
19         this.tahunProduksi = tahunProduksi;
20     }
21
22     // Method untuk mendapatkan informasi elektronik
23     public String getInfo() {
24         return "Merk      : " + merk + "\n" +
25             "Model      : " + model + "\n" +
26             "Tahun Produksi: " + tahunProduksi + "\n";
27     }
28
29
30
31
32
```

### b. Kelas Televisi



```
Welcome | Elektronik11.java | Televisi11.java X | Smartphone11.java | Demo11.java
Percobaan > J Televisi11.java > Televisi11 > Televisi11(String, String, int, String)
1 package Percobaan;
2
3 public class Televisi11 extends Elektronik11 {
4     // Atribut tambahan
5     private String ukuranLayar;
6
7     public Televisi11() {
8         super();
9         this.ukuranLayar = "N/A";
10    }
11
12    // Constructor berparameter
13    public Televisi11(String merk, String model, int tahunProduksi, String ukuranLayar) {
14        super(merk, model, tahunProduksi);
15        this.ukuranLayar = ukuranLayar;
16    }
17
18    // Method untuk mendapatkan informasi televisi
19    @Override
20    public String getInfo() {
21        return super.getInfo() + "Ukuran Layar : " + ukuranLayar + "\n";
22    }
23
24    // Getter untuk ukuranLayar
25    public String getUkuranLayar() {
26        return ukuranLayar;
27    }
28
29    // Setter untuk ukuranLayar
30    public void setUkuranLayar(String ukuranLayar) {
31        this.ukuranLayar = ukuranLayar;
32    }
33 }
34
```

c. Kelas smartphone

```
Welcome | Elektronik11.java | Televisi11.java | Smartphone11.java X | Demo11.java
Percobaan > J Smartphone11.java > Smartphone11 > Smartphone11(String, String, int, int)
1 package Percobaan;
2
3 public class Smartphone11 extends Elektronik11 {
4     private int kapasitasBaterai; // dalam mAh
5
6     // Constructor tanpa parameter
7     public Smartphone11() {
8         super();
9         this.kapasitasBaterai = 0;
10    }
11    public Smartphone11(String merk, String model, int tahunProduksi, int kapasitasBaterai) {
12        super(merk, model, tahunProduksi);
13        this.kapasitasBaterai = kapasitasBaterai;
14    }
15
16    // Method untuk mendapatkan informasi smartphone
17    @Override
18    public String getInfo() {
19        return super.getInfo() + "Kapasitas Baterai: " + kapasitasBaterai + " mAh\n";
20    }
21
22    // Getter untuk kapasitasBaterai
23    public int getKapasitasBaterai() {
24        return kapasitasBaterai;
25    }
26
27    // Setter untuk kapasitasBaterai
28    public void setKapasitasBaterai(int kapasitasBaterai) {
29        this.kapasitasBaterai = kapasitasBaterai;
30    }
31 }
32
```

d. Kelas demo

```
Welcome | Elektronik11.java | Televisi11.java | Smartphone11.java | Demo11.java X
Percobaan > J Demo11.java > Demo11 > main(String[])
1 package Percobaan;
2
3 public class Demo11 {
4     public static void main(String[] args) {
5         Televisi11 tv1 = new Televisi11(merk:"Samsung", model:"QLED", tahunProduksi:2022, ukuranLayar:"55 inci");
6
7         System.out.println(x:"Informasi Televisi:");
8         System.out.println(tv1.getInfo());
9
10        // Modifikasi nilai atribut
11        tv1.setUkuranLayar(ukuranLayar:"65 inci");
12
13        System.out.println(x:"Informasi Televisi setelah modifikasi:");
14        System.out.println(tv1.getInfo());
15
16        // Instansiasi objek Smartphone menggunakan constructor berparameter
17        Smartphone11 smartphone1 = new Smartphone11(merk:"Apple", model:"iPhone 15", tahunProduksi:2023, kapasitasBaterai:3279);
18
19        // Menampilkan informasi Smartphone
20        System.out.println(x:"Informasi Smartphone:");
21        System.out.println(smartphone1.getInfo());
22
23        smartphone1.setKapasitasBaterai(kapasitasBaterai:3500);
24
25        // Menampilkan informasi setelah modifikasi
26        System.out.println(x:"Informasi Smartphone setelah modifikasi:");
27        System.out.println(smartphone1.getInfo());
28    }
29 }
```

e. Hasil Rum

```
PS D:\SEMESTER 3\PRO\Java\Jobsheet07> d:; cd 'd:\SEMESTER 3\PRO\Java\Jobsheet07';
del6\redhat.java\jdt_ws\Jobsheet07_4adc1d9\bin\ 'Percobaan.Demo11'
Informasi Televisi:
Merk      : Samsung
Model     : QLED
Tahun Produksi: 2022
Ukuran Layar : 55 inci

Informasi Televisi setelah modifikasi:
Merk      : Samsung
Model     : QLED
Tahun Produksi: 2022
Ukuran Layar : 65 inci

Informasi Smartphone:
Merk      : Apple
Model     : iPhone 15
Tahun Produksi: 2023
Kapasitas Baterai: 3279 mAh

Informasi Smartphone setelah modifikasi:
Merk      : Apple
Model     : iPhone 15
Tahun Produksi: 2023
Kapasitas Baterai: 3500 mAh

PS D:\SEMESTER 3\PRO\Java\Jobsheet07> |
```

**Getters and Setters:** Getters are used to get the value of a private attribute, and setters are used to change the value of that attribute. This follows the principle of encapsulation in object-oriented programming.

**Using Setters:** In the Demo class, we can now use setters to modify the values of the ScreenSize and BatteryCapacity attributes.

--- selamat mengerjakan----

