

JOBSHEET W06

INHERITANCE

1. COMPETENCE

1. Understand the basic concept of inheritance.
2. Able to create a subclass of a certain superclass.
3. Able to implement the concept of hierarchical inheritance
4. Able to create objects from a subclass and access attributes and methods either own or derived from their superclass.

2. INTRODUCTION

Inheritance in object oriented programming is the concept of **inheritance** from a more general class to a more specific class. The class that is derived is called the base class (**base class/super class/parent class**), while the class that is derived is called a derived **class (sub class/child class)**. Each **subclass** will "inherit" the attributes and methods of the public or protected *superclass*. The benefit of inheritance is *reusability* or reuse of lines of code.

In the Java programming language, inheritance declarations are made by adding the **extends** keyword after the class name declaration, followed by the parent class--name. The extends keyword tells the Java compiler that we want to do **an extension/extension** of the class. Here is an example of an inheritance declaration.

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

The example above tells the Java compiler that class B is extending class A. This means that class B is a subclass of class A by extension. This extension will be done by adding special attributes and methods that only class B has.

A parent class can limit the attributes and methods that will be inherited to its subclasses. The restriction is carried out through the determination of access level modifiers. In Java, the access level modifier attributes and methods are summarized in the following table:

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

Attributes and methods that will be inherited from parent class to child class are attributes and methods with a protected or public modifier.

The keyword **this** is used to refer to the current object/class. While the **super** keyword is used to refer to the parent object/class. The writing format is as follows:

- **super.<nameAttributes>**
Accessing parent attributes
- **super.<nameMethod>()**

Calling the parent method

1. EXPERIMENT 1 (extends)

A. TRIAL STAGES

1. Create a parent class with the name of the Pegawai. Then create a parameterless constructor with the following line of code:

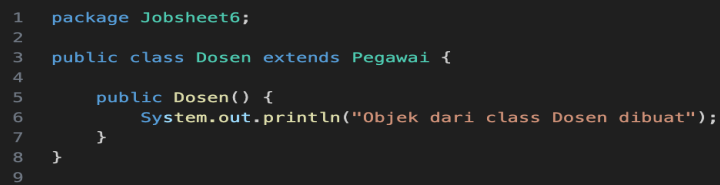
```
public class Pegawai {  
  
    public Pegawai() {  
        System.out.println("Objek dari class Pegawai dibuat");  
    }  
  
}
```



```
1 package Jobsheet6;  
2  
3 public class Pegawai {  
4  
5     public Pegawai() {  
6         System.out.println("Objek dari class Pegawai dibuat");  
7     }  
8 }  
9
```

2. Create a subclass of the Pegawai class with the name Dosen, then also create a parameterless constructor with the following line of code:

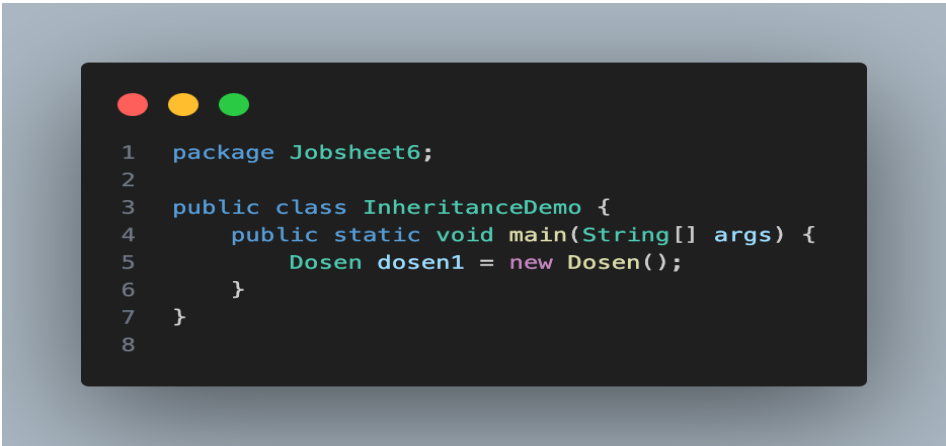
```
public class Dosen extends Pegawai {  
  
    public Dosen() {  
        System.out.println("Objek dari class Dosen dibuat");  
    }  
  
}
```



```
1 package Jobsheet6;  
2  
3 public class Dosen extends Pegawai {  
4  
5     public Dosen() {  
6         System.out.println("Objek dari class Dosen dibuat");  
7     }  
8 }  
9
```

3. Create a main class, for example InheritanceDemo.java, instantiate a new object named dosen1 from the lecturer class as follows:

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




```

1  package Jobsheet6;
2
3  public class InheritanceDemo {
4      public static void main(String[] args) {
5          Dosen dosen1 = new Dosen();
6      }
7  }
8

```

4. Run the program and then observe the results.



```

Objek dari class Pegawai dibuat
Objek dari class Dosen dibuat
nazril@Muhammads-MacBook-Air Jobsheet %

```

Screenshot by Xnapper.com

B. QUESTION

1. In experiment 1 above, determine the child class and parent class!
 - Parent class: Pegawai
 - Child class: Dosen
2. What keywords make the child class and parent class have a relationship?
 - Kata kunci extends digunakan untuk menunjukkan bahwa class Dosen adalah subclass (child class) dari class Pegawai (parent class).
3. Based on the results displayed by the program, how many constructors are executed? Which constructor class is executed first?
 - Ada dua constructor yang dieksekusi. Pertama, constructor dari class Pegawai, lalu constructor dari class Dosen.
 - Constructor class Pegawai dieksekusi lebih dulu karena dalam Java, constructor dari parent class selalu dipanggil terlebih dahulu sebelum constructor dari child class saat membuat objek dari child class.

4. EXPERIMENT 2 (Inheritance)

A. TRIAL STAGES

1. Add the nip, nama, and gaji attributes and the getInfo() method to the Pegawai class

```
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 Jobsheet6;  
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  
12     public String getInfo() {  
13         String info = "";  
14         info += "NIP      : " + nip + "\n";  
15         info += "Nama      : " + nama + "\n";  
16         info += "Gaji      : " + gaji + "\n";  
17         return info;  
18     }  
19 }  
20
```

2. Also add the NIDN attribute to the Dosen class

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

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



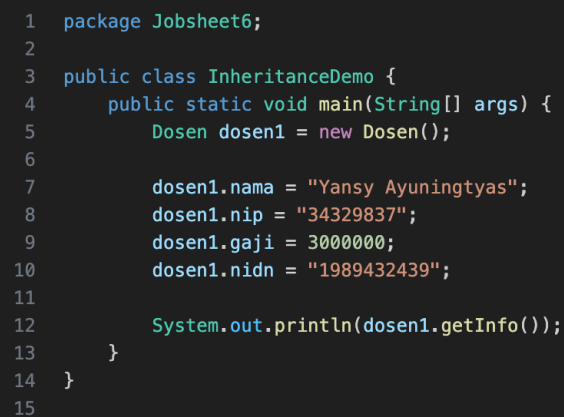
```
1 package Jobsheet6;
2
3 public class Dosen extends Pegawai {
4     public String nidn;
5
6     public Dosen() {
7         System.out.println("Objek dari class Dosen dibuat");
8     }
9 }
10
```

3. In class InheritanceDemo.java write the following line of code:

```
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 Jobsheet6;
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 the program then observe the results



```
Objek dari class Pegawai dibuat
Objek dari class Dosen dibuat
NIP      : 34329837
Nama     : Yansy Ayuningtyas
Gaji     : 3000000.0
nazril@Muhammads-MacBook-Air Jobsheet %
```

Screenshot by Xnapper.com

B. QUESTION

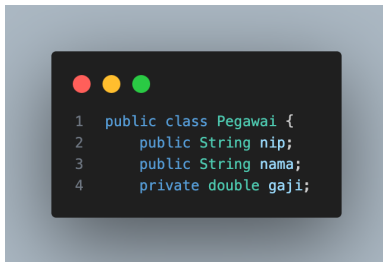
1. In experiment 2 above, can the program run successfully or does an error occur?
 - Program berhasil dijalankan tanpa error. Hal ini terlihat dari output yang dihasilkan dengan informasi dari objek dosen yang mencakup NIP, Nama, dan Gaji.
2. If the program is successfully executed, why is there no error in the assignment/filling in the values of the nip, gaji, and NIDN attributes on the lecturer object1 even though there is no declaration of these three attributes in the lecturer class?
 - Tidak terjadi error karena atribut nip dan gaji merupakan atribut warisan (inherited) dari class Pegawai. Dalam konsep inheritance (pewarisan), subclass (Dosen) mewarisi semua atribut dan method yang bersifat public atau protected dari parent class (Pegawai), sehingga nip dan gaji yang dideklarasikan di class Pegawai otomatis dapat diakses dan digunakan oleh objek dosen .
 - Sedangkan, atribut nidn dideklarasikan langsung di class Dosen, sehingga bisa langsung diakses oleh objek dosen.
3. If the program is successfully executed, why is there no error in the call of the getInfo() method by the lecturer1 object even though there is no getInfo() method declaration in the dosen class?
 - Tidak terjadi error karena method getInfo() dideklarasikan di parent class (Pegawai), dan sesuai dengan konsep inheritance, subclass (Dosen) juga mewarisi semua method yang bersifat **public** dari parent class. Oleh karena itu, objek dosen1 dapat memanggil method getInfo() meskipun tidak ada deklarasi khusus method tersebut dalam class Dosen.

5. EXPERIMENT 3 (Access rights)

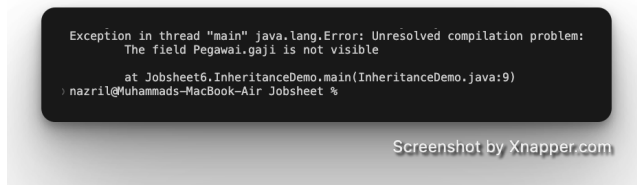
A. TRIAL STAGES

1. Modification of access level modifier on gaji attributes to private in class Pegawai.java

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



2. Run the program then observe the results.



3. Change the access level modifier of the gaji attribute to protected and then move the Pegawai class to a new package, for example "testpackage".

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



4. Import the Pegawai class from the testpackage in the Dosen class.

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



5. Access salary attributes in the Dosen class by trying to print the gaji attributes in the Lecturer constructor

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



6. Change the access level modifier back to public and revert the Pegawai class to the original package.



B. QUESTION

1. In step 1 above, an error occurred because the dosen object1 could not access the gaji attributes. Even though gaji is an attribute of an pegawai who is the parent class of the dosen. Why does this happen?

The error occurs because the gaji attribute is marked **private**. In OOP, a **private** attribute can only be accessed within its own class and not by subclass or other objects. Hence, dosen1 cannot directly access gaji, even though it's inherited from the parent class Pegawai

2. In step 5, after the Pegawai class moves to a different package, the Dosen class can still access the gaji attributes. Why?

After changing the gaji access modifier to **protected**, the attribute becomes accessible to **subclasses**, even if they are in different packages. The **protected** access level allows subclass access to inherited attributes across packages.

3. Based on the experiment, how to determine the attributes and methods that will be inherited by the parent class to the child class?

Inheritance depends on the **access modifiers**:

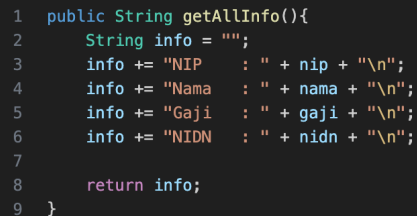
- **public**: Accessible by all classes, including subclasses in different packages.
- **protected**: Accessible by subclasses, even across packages.
- **default** (no modifier): Accessible within the same package.
- **private**: Not accessible outside the class where it's defined, including subclasses.

6 . EXPERIMENT 4 (Super - attributes)

A. TRIAL STAGES

1. Use the getAllInfo() method in the Dosen class

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



```
1 public String getAllInfo(){  
2     String info = "";  
3     info += "NIP      : " + nip + "\n";  
4     info += "Nama      : " + nama + "\n";  
5     info += "Gaji      : " + gaji + "\n";  
6     info += "NIDN      : " + nidn + "\n";  
7  
8     return info;  
9 }
```

2. Call the getAllInfo() method by the dosen1 object on 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());  
}
```



```
1 package Jobsheet6;  
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).getAllInfo());  
13    }  
14 }  
15 }  
16 }
```

3. Run the program then observe the results



4. Modify the `getAllInfo()` method in the `Dosen` class

```
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;  
}
```



5. Run the program then compare the results with step no 2.
6. Modify the `getAllInfo()` method on the `Dosen` class again

```
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;  
}
```

```

1
2 public String getAllInfo(){
3     String info = "";
4     info += "NIP      : " + super.nip + "\n";
5     info += "Nama      : " + super.nama + "\n";
6     info += "Gaji      : " + super.gaji + "\n";
7     info += "NIDN      : " + super.nidn + "\n";
8
9     return info;
10 }

```

7. Run the program and then compare the results with the program at no 1 and no 4.

```

n Jobsheet6\InheritanceDemo
Objek dari class Pegawai dibuat
0.0
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 Jobsheet6.Dosen.getAllInfo(Dosen.java:16)
    at Jobsheet6.InheritanceDemo.main(InheritanceDemo.java:12)
nazril@Muhammads-MacBook-Air Jobsheet %

```

Screenshot by Xnapper.com

8. Modify the getAllInfo() method on the Dosen class again

```

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;
}

```

```

1 public String getAllInfo(){
2     String info = "";
3     info += "NIP      : " + super.nip + "\n";
4     info += "Nama      : " + super.nama + "\n";
5     info += "Gaji      : " + super.gaji + "\n";
6     info += "NIDN      : " + this.nidn + "\n";
7
8     return info;
9 }

```

- Run the program and then compare the results with the program at number 2 and number 4.



B. QUESTION

- Are there any differences in the results of nama, nip, and gaji displayed in programs 1, 4, and 8? Why?
nothing
- Why does the error occur in program no 6?
super.nidn causes an error because nidn does not exist in the parent class Employee.
this.nidn is used because nidn is an attribute of the Lecturer class, and this refers to the class instance itself.

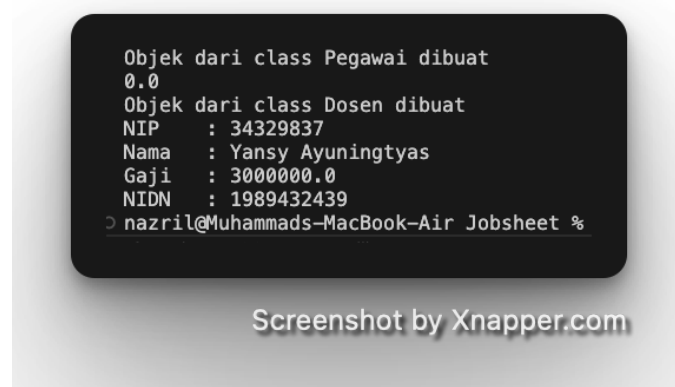
7. EXPERIMENT 5 (super & overriding)

A. TRIAL STAGES

- Modify the getAllInfo() method again. Run the program then observe the results

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

    return info;
}
```



- Modify the getAllInfo() method again. Run the program then observe the results

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

    return info;
}
```

```

Objek dari class Pegawai dibuat
0.0
Objek dari class Dosen dibuat
NIP : 34329837
Nama : Yansy Ayuningtyas
Gaji : 3000000.0
NIDN : 1989432439
nazril@Muhammads-MacBook-Air Jobsheet %

```

Screenshot by Xnapper.com

3. Modify the `getAllInfo()` method again. Run the program then observe the results

```

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

    return info;
}

```

```

// Jobsheet 1: Inheritance Demo
Objek dari class Pegawai dibuat
0.0
Objek dari class Dosen dibuat
NIP : 34329837
Nama : Yansy Ayuningtyas
Gaji : 3000000.0
NIDN : 1989432439
nazril@Muhammads-MacBook-Air Jobsheet %

```

Screenshot by Xnapper.com

4. Add the `getInfo()` method to the `Dosen` class and modify the `getAllInfo()` method as follows

```

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;
    }
}

```

```

.....
Objek dari class Pegawai dibuat
Objek dari class Dosen dibuat
NIP      : 34329837
Nama     : Yansy Ayuningtyas
Gaji     : 3000000.0
NIDN     : 1989432439
nazril@Muhammads-MacBook-Air Jobsheet %

```

Screenshot by Xnapper.com

B. QUESTION

- Are there any differences in the `getInfo()` methods accessed in steps 1, 2, and 3?
NO
- Is there a difference between the `super.getInfo()` and `this.getInfo()` methods called in the `getAllInfo()` method in step 4? Explain!
 - `super.getInfo()`**: Calls the `getInfo()` method from the parent class **Pegawai**, which displays **NIP, Name, and Salary**.
 - `this.getInfo()`**: Calls the `getInfo()` method from the **Dosen** class (if overridden), which in this case only displays **NIDN**.
- In what method does overriding occur? Explain!
Overriding occurs when a subclass provides a new implementation for a method that already exists in its superclass. In the provided code example, **overriding** happens in the `getInfo()` method within the **Dosen** class.

8. EXPERIMENT 6 (overloading)

A. TRIAL STAGES

- Add a new constructor for the `Dosen` class as follows

```

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

```

- Modify the `InheritanceDemo` class to instantiate a new object with the name `lecturer2` with a parameterized constructor. Run the program then observe the results.

```

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

```

```

Objek dari class Dosen dibuat dengan constructor
NIP      : 34329837
Nama     : Yansy Ayuningtyas
Gaji     : 3000000.0
NIDN     : 1989432439
nazril@Muhammads-MacBook-Air Jobsheet %

```

Screenshot by Xnapper.com

B. QUESTION

1. What are the results of the nip, nama, gaji, and nidn values displayed in step 2? Why is that?
- **NIP, Nama, and Gaji** are assigned in the Dosen constructor from the parameters passed when creating the object (new Dosen("34329837", "Yansy Ayuningtyas", 3000000, "1989432439")). These attributes are inherited from the superclass **Pegawai**.
- **NIDN** is specific to the **Dosen** class and is also assigned from the constructor parameter.
2. Explain whether the parameterless constructor and the Dosen class constructor created in step 1 have the same signature?
No, the parameterless constructor and the Dosen class constructor do not have the same signature. The signatures are different because one has no parameters, while the other has four. In Java, constructors are distinguished by the number and types of their parameters.
3. What is the concept in OOP that allows a class to have a constructor or method with the same name and a different signature on a class?

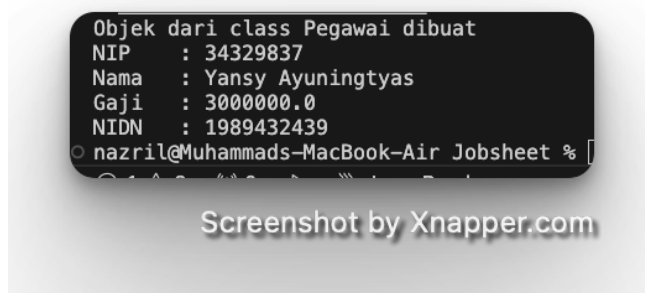
The concept in OOP that allows a class to have a constructor or method with the same name but different signatures is called **method overloading**.

9. EXPERIMENT 7 (super-constructor)

A. TRIAL STAGES

1. Constructor modifications in the Dosen class are as follows. Run the program then observe the results.

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



2. Constructor modifications in the Dosen class are as follows. Run the program then observe the results.

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

```

Objek dari class Pegawai dibuat
NIP      : 34329837
Nama     : Yansy Ayuningtyas
Gaji     : 3000000.0
NIDN     : 1989432439
nazril@Muhammads-MacBook-Air Jobsheet %

```

Screenshot by Xnapper.com

3. Constructor modifications in the Dosen class are as follows. Run the program then observe the results.

```

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

```

```

Objek dari class Pegawai dibuat
NIP      : 34329837
Nama     : Yansy Ayuningtyas
Gaji     : 3000000.0
NIDN     : 1989432439
nazril@Muhammads-MacBook-Air Jobsheet %

```

Screenshot by Xnapper.com

4. Remove/comment constructor without parameters from the Pegawai class. Add a new constructor for the Pegawai class as follows. Run the program then observe the results.

```

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;
    }
}

```


5. Constructor modifications in the Dosen class are as follows. Run the program then observe the results.

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

```
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 Jobsheet6.Dosen.<init>(Dosen.java:46)
at Jobsheet6.InheritanceDemo.main(InheritanceDemo.java:16)
nazril@Muhammads-MacBook-Air Jobsheet %
```

Screenshot by Xnapper.com

6. Constructor modifications in the Dosen class are as follows. Run the program then observe the results.

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

```
NIP : 34329837
Nama : Yansy Ayuningtyas
Gaji : 3000000.0
NIDN : 1989432439
nazril@Muhammads-MacBook-Air Jobsheet %
```

Screenshot by Xnapper.com

B. QUESTION

1. Is there a difference in the results in steps 1 and 2? Explain!

Yes, there is a difference. Step 1 with the no-argument constructor yields default values (null for strings, 0.0 for doubles), while step 2 with the parameterized constructor displays values according to the provided input.

2. Is there a difference in the results in steps 2 and 3? Explain!

There is no difference in results. Both use the same constructor, so the output should be the same, displaying the values of nip, nama, gaji, and nidn.

3. Why did the error occur in step 4?

The error occurs because `super()` attempts to call the default constructor `Pegawai()`, which does not exist. It should use `super(nip, nama, gaji)` for the parameterized constructor.

4. What is the difference between `super()` called in steps 3 and 6?

- **`super()` in step 3:** Calls the default constructor that does not exist.
- **`super(nip, nama, gaji)` in step 6:** Calls the valid parameterized constructor of `Pegawai`, used for initialization.

5. Why did the error occur in step 5?

The error occurs because the required parameters for the `Dosen` constructor are not fulfilled. Ensure to provide all necessary parameters when creating the object.

10. ASSIGNMENT

1. Define a class that is a derivative of another class.
2. Create 3 attributes in the parent class then add at least 1 attribute in the child class.
3. Perform the overloading method by creating 2 constructors, namely a parameterless constructor and a parameterized constructor for each class. Call the parameterized super() constructor to create an object from the parent class on the child class constructor.
4. Implement the class diagram made in the theoretical PBO course
5. Create a Demo class then instantiate the child class object in the main function
6. Try modifying the attribute values (both those declared in the child class and those inherited from the print info.

ANSWER:

```
1  package Jobsheet6;
2
3  public class PerangkatElektronik {
4      private String merk;
5      private String mode;
6      private int tahunProduksi;
7
8      public PerangkatElektronik() {
9          this.merk = "Unknown";
10         this.mode = "Unknown";
11         this.tahunProduksi = 0;
12     }
13
14     public PerangkatElektronik(String merk, String mode, int tahunProduksi) {
15         this.merk = merk;
16         this.mode = mode;
17         this.tahunProduksi = tahunProduksi;
18     }
19
20     public void displayInfo() {
21         System.out.println("Merk: " + merk);
22         System.out.println("Mode: " + mode);
23         System.out.println("Tahun Produksi: " + tahunProduksi);
24     }
25
26     public String getMerk() {
27         return merk;
28     }
29
30     public String getMode() {
31         return mode;
32     }
33
34     public int getTahunProduksi() {
35         return tahunProduksi;
36     }
37 }
38
```



```
1 package Jobsheet6;
2
3 public class TV extends PerangkatElektronik {
4     private int ukuranLayar;
5
6     public TV() {
7         super();
8         this.ukuranLayar = 0;
9     }
10
11     public TV(String merk, String mode, int tahunProduksi, int ukuranLayar) {
12         super(merk, mode, tahunProduksi);
13         this.ukuranLayar = ukuranLayar;
14     }
15
16     public void displayInfoTV() {
17         displayInfo();
18         System.out.println("Ukuran Layar: " + ukuranLayar + " inci");
19     }
20 }
21
```



```
1 package Jobsheet6;
2
3 public class Tablet extends PerangkatElektronik {
4     private int kapasitasBaterai;
5
6     public Tablet() {
7         super();
8         this.kapasitasBaterai = 0;
9     }
10
11     public Tablet(String merk, String mode, int tahunProduksi, int kapasitasBaterai) {
12         super(merk, mode, tahunProduksi);
13         this.kapasitasBaterai = kapasitasBaterai;
14     }
15
16     public void displayInfoTablet() {
17         displayInfo();
18         System.out.println("Kapasitas Baterai: " + kapasitasBaterai + " mAh");
19     }
20 }
21
```

```

1  package Jobsheet6;
2
3  public class Demo {
4      public static void main(String[] args) {
5          System.out.println("");
6          TV tv = new TV("Samsung", "Model A", 2022, 55);
7          System.out.println("TV Information:");
8          System.out.println("-----");
9          tv.displayInfoTV();
10
11         System.out.println();
12
13         Tablet tablet = new Tablet("Apple", "iPad", 2021, 5000);
14         System.out.println("Tablet Information:");
15         System.out.println("-----");
16         tablet.displayInfoTablet();
17         System.out.println("");
18     }
19 }
20

```

TV Information:

Merk: Samsung
 Mode: Model A
 Tahun Produksi: 2022
 Ukuran Layar: 55 inci

Tablet Information:


Merk: Apple
 Mode: iPad
 Tahun Produksi: 2021
 Kapasitas Baterai: 5000 mAh

nazril@Muhammads-MacBook-Air Jobsheet %

Screenshot by Xnapper.com


modifying the attribute values

- Class PerangkatElektronik



```
1  //Modif
2  public void setMerk(String merk) {
3      this.merk = merk;
4  }
5
6  public void setMode(String mode) {
7      this.mode = mode;
8  }
9
10 public void setTahunProduksi(int tahunProduksi) {
11     this.tahunProduksi = tahunProduksi;
12 }
```

- Class TV



```
1  //modif
2  public int getUkuranLayar() {
3      return ukuranLayar;
4  }
5
6  public void setUkuranLayar(int ukuranLayar) {
7      this.ukuranLayar = ukuranLayar;
8  }
```

- Class Tablet



```
1 //modif
2 public int getKapasitasBaterai() {
3     return kapasitasBaterai;
4 }
5
6 public void setKapasitasBaterai(int kapasitasBaterai) {
7     this.kapasitasBaterai = kapasitasBaterai;
8 }
```

- Main Demo



```
1 //modif
2 tv.setMerk("LG");
3 tv.setMode("Model B");
4 tv.setTahunProduksi(2023);
5 tv.setUkuranLayar(65);
6
7 System.out.println("\nAfter modification:");
8 tv.displayInfoTV();
```



```
1
2 //modif
3 tablet.setMerk("Samsung");
4 tablet.setMode("Galaxy Tab");
5 tablet.setTahunProduksi(2022);
6 tablet.setKapasitasBaterai(8000);
7
8 System.out.println("\nAfter modification:");
9 tablet.displayInfoTablet();
```

TV Information:

Merk: Samsung
Mode: Model A
Tahun Produksi: 2022
Ukuran Layar: 55 inci

After modification:
Merk: LG
Mode: Model B
Tahun Produksi: 2023
Ukuran Layar: 65 inci

Tablet Information:

Merk: Apple
Mode: iPad
Tahun Produksi: 2021
Kapasitas Baterai: 5000 mAh

After modification:
Merk: Samsung
Mode: Galaxy Tab
Tahun Produksi: 2022
Kapasitas Baterai: 8000 mAh

○ nazril@Muhammads-MacBook-Air Jobsheet % █

Screenshot by Xnapper.com

--- happy working----