

# SAFRIZAL RAHMAN 19 SIB 2G

## LINK

[https://github.com/safrizalrahman46/PBO\\_SAFRIZ\\_THEVIGILA](https://github.com/safrizalrahman46/PBO_SAFRIZ_THEVIGILA)

NTE

### 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 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:

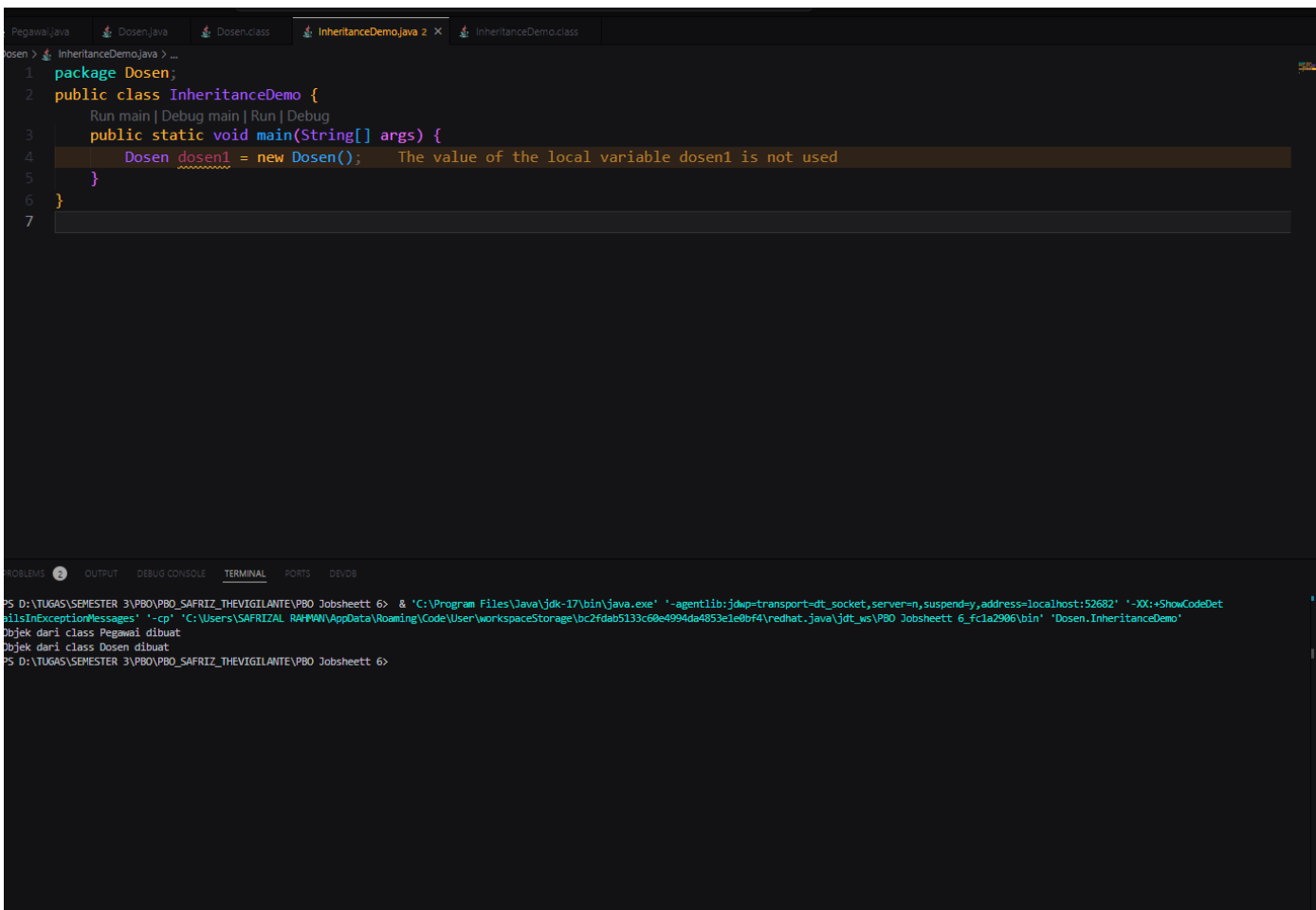
```
public class Dosen extends Pegawai {

    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();
}
```



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

PS D:\TUGAS\SEMESTER 3\PBO\PBO\_SAFRIZ\_THEVIGILANTE\PBO Jobsheett 6> & 'C:\Program Files\Java\jdk-17\bin\java.exe' "-agentlib:jdwp=transport=dt\_socket,server=n,suspend=y,address=localhost:52682" "-XX:+ShowCodeDet

4. Run programnya kemudian amati hasilnya.

## B. PERTANYAAN

1. Pada percobaan 1 diatas, tentukan child class dan parent class!

- **Child class:** Dosen
- **Parent class:** Pegawai

2. Kata kunci apa yang membuat child class dan parent class tersebut memiliki relasi?

Kata kunci yang digunakan untuk membuat relasi antara child class dan parent class adalah kata kunci `extends`. Ini menunjukkan bahwa Dosen mewarisi properti dan metode dari Pegawai.

3. Berdasarkan hasil yang ditampilkan oleh program, ada berapa constructor yang dieksekusi? Constructor class mana yang lebih dulu dieksekusi?

Ada dua constructor yang dieksekusi:

- Constructor dari Pegawai (Parent class)
- Constructor dari Dosen (Child class)

Constructor dari **class Pegawai** dieksekusi terlebih dahulu, diikuti oleh constructor dari class Dosen. Ini karena ketika sebuah objek subclass (Dosen) dibuat, constructor dari parent class (Pegawai) secara otomatis dipanggil terlebih dahulu.

## 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;
    }
}
```

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");
    }
}
```

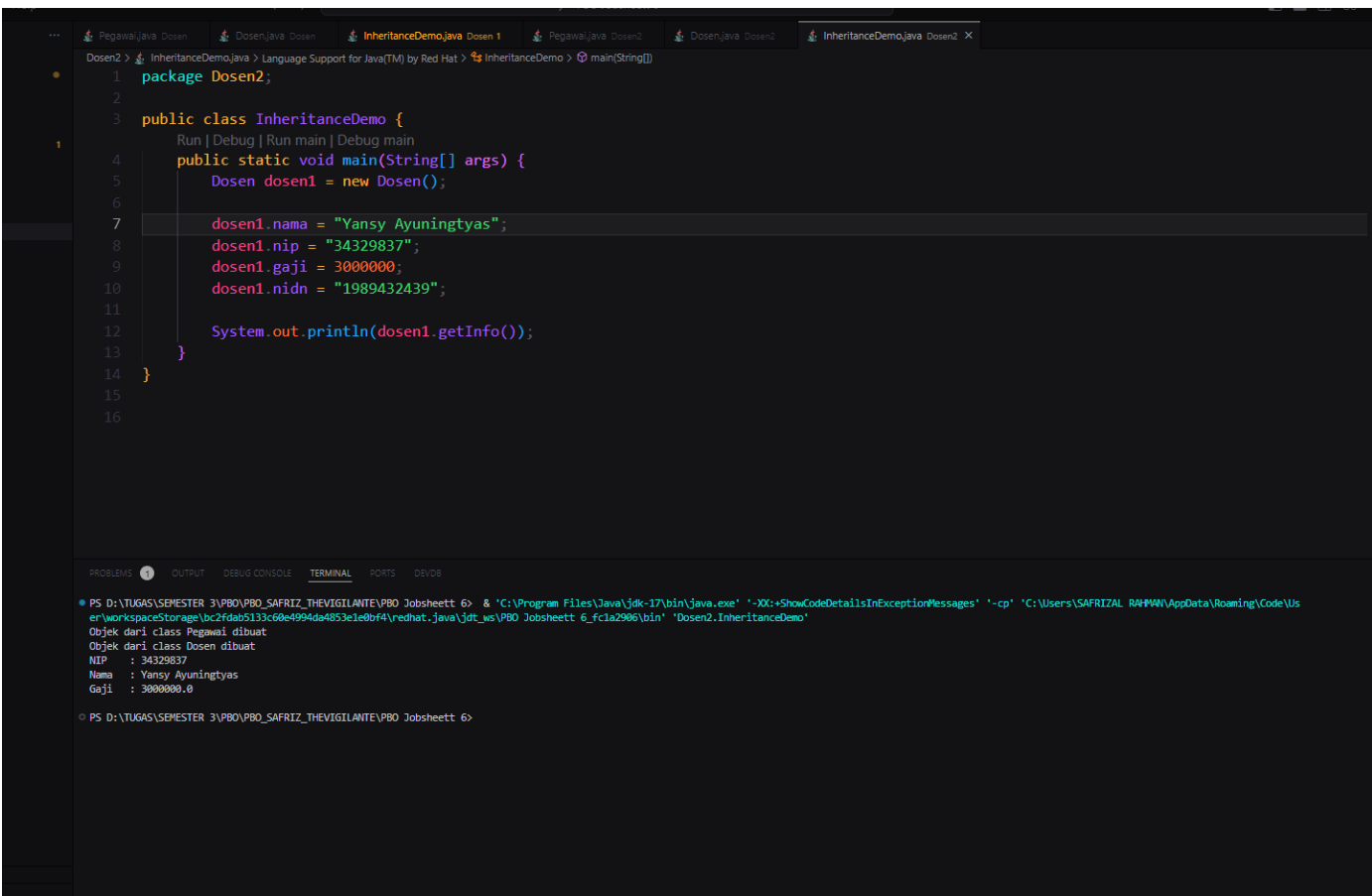
3. Pada class InheritanceDemo.java tuliskan baris kode berikut:

```
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());
}
```

4. Run program kemudian amati hasilnya



```
package Dosen2;

public class InheritanceDemo {
    Run | Debug | Run main | Debug main
    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());
    }
}
```

PROBLEMS 1 OUTPUT DEBUG CONSOLE TERMINAL PORTS DEVICES

PS D:\TUGAS\SEMESTER 3\PBO\PBO\_SAFRIZ\_THEVIGILANTE\PBO Jobsheett 6> & 'C:\Program Files\Java\jdk-17\bin\java.exe' '-XX:+ShowCodeDetailsInExceptionMessages' '-cp' 'C:\Users\SAFRIZAL\_RAHMAN\AppData\Roaming\Code\Use...er\workspaceStorage\bc2fdab5133c60e4994da4853e1e0bf4\redhat-java\jdk\_ws\PBO Jobsheett 6\_fc1a2906\bin' 'Dosen2.InheritanceDemo'

Objek dari class Pegawai dibuat  
Objek dari class Dosen dibuat  
NIP : 34329837  
Nama : Yansy Ayuningtyas  
Gaji : 3000000.0

PS D:\TUGAS\SEMESTER 3\PBO\PBO\_SAFRIZ\_THEVIGILANTE\PBO Jobsheett 6>

## B. PERTANYAAN

1. Pada percobaan 2 diatas, apakah program dapat berhasil dijalankan atautkah terjadi error?  
Program **berhasil dijalankan** tanpa error, karena class Dosen secara otomatis mewarisi semua atribut dan method yang ada di class Pegawai.
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?  
Tidak terjadi error karena atribut nip, nama, dan gaji dideklarasikan di class Pegawai, dan class Dosen mewarisi semua atribut dari class Pegawai menggunakan keyword extends. Dengan kata lain, atribut tersebut tersedia dalam objek dosen1, yang merupakan instance dari class Dosen, tetapi atribut tersebut diwarisi dari class Pegawai.
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?  
Tidak terjadi error karena method getInfo() dideklarasikan di class Pegawai, dan class Dosen mewarisi method tersebut. Karena Dosen merupakan subclass dari Pegawai, semua method public atau protected dari class Pegawai tersedia untuk objek dosen1, sehingga getInfo() dapat dipanggil dari objek dosen1 tanpa masalah.

## 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;  
}
```

Import class Pegawai dari testpackage pada class Dosen.

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

- 4.
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.

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

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS DEVDS

```
PS D:\TUGAS\SEMESTER 3\PROJ\PROJ_SAFRIZ_THEVIGILANTE\PROJ Jobsheet 6> java -cp "C:\Program Files\Java\jdk-17\bin\java.exe" "-Xt:-ShowCodeDetailsInExceptionMessages" "-cp" "C:\Users\SAFRIZAL_RAPMAN\AppData\Roaming\Code\Us
er\workspaceStorage\1bc2f4a5133c684994a483e1a8b74\workspace\java\jdt_ws\PROJ Jobsheet 6_fc1a2980\bin" "Inheritance.InheritanceDemo"
Objek dari class Pegawai dibuat
Gaji: 0.0
Objek dari class Dosen dibuat
NIP : 34329837
Nama : Yansy Ayuningtyas
Gaji : 3000000.0
```

## 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?  
Hal ini terjadi karena akses modifier **private** membatasi akses atribut gaji hanya di dalam class Pegawai itu sendiri. Meskipun Dosen adalah subclass dari Pegawai, atribut yang bersifat **private** tidak dapat diwariskan atau diakses langsung oleh subclass. Hanya metode dari class Pegawai yang dapat mengakses atribut gaji (misalnya melalui getter).
2. Pada langkah 5, setelah class Pegawai berpindah ke package yang berbeda, class Dosen masih dapat mengakses atribut gaji. Mengapa?  
Setelah access modifier diubah menjadi **protected**, atribut gaji menjadi dapat diakses oleh class yang merupakan subclass, meskipun Dosen berada di package yang berbeda. Modifier **protected** memungkinkan akses ke atribut di subclass, baik di dalam package yang sama maupun package yang berbeda, selama ada hubungan pewarisan.
3. Berdasarkan percobaan tersebut, bagaimana menentukan atribut dan method yang akan diwariskan oleh parent class ke child class?

a **Public:** Atribut dan method dengan modifier **public** dapat diakses dari mana saja, termasuk dari subclass dan package lain.

b **Protected:** Atribut dan method dengan modifier **protected** dapat diakses oleh subclass, meskipun berada di package yang berbeda, tetapi tidak oleh class di luar package yang tidak memiliki hubungan pewarisan.

c **Default/Package-private:** Jika tidak ada modifier yang diberikan (akses default), atribut dan method hanya bisa diakses oleh class lain dalam package yang sama.

d **Private:** Atribut dan method dengan modifier **private** hanya bisa diakses dalam class yang mendeklarasikannya dan tidak bisa diwariskan atau diakses oleh subclass.

Jadi, ketika merancang class, jika kita ingin agar subclass dapat mengakses atribut atau method, kita bisa menggunakan **protected** atau **public**. Namun, jika atribut tersebut tidak perlu diakses langsung, kita bisa menggunakan **private** dan menyediakan method getter/setter untuk mengontrol aksesnya.

## 6. PERCOBAAN 4 (Super - atribut)

### A. TAHAPAN PERCOBAAN

1. Buatlah 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;  
}
```

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());  
}
```

3. Run program kemudian amati hasilnya
4. 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;  
}
```

5. Run program kemudian bandingkan hasilnya dengan langkah no 2.
6. 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;
}

```

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

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;
}

```

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

SEBELUM

THIS

The screenshot shows an IDE with the following tabs: Pegawai.java Dosen, Dosen.class, Dosen.java Dosen, InheritanceDemo.java Dosen 2, and Pegawai.class. The main editor displays the code for InheritanceDemo.java, which is part of the package Dosen2. The code defines a public class InheritanceDemo with a main method. Inside the main method, a Dosen object named dosen1 is created and its attributes (nama, nip, gaji, nidn) are set. The code then calls System.out.println(dosen1.getInfo()); and System.out.println(dosen1.getAllInfo());. The IDE shows a lightbulb icon next to the first println statement, indicating a suggestion or warning. The terminal at the bottom shows the output of the program, which is the result of running the main method. The output shows the details of the Dosen object dosen1, including its NIP, Nama, Gaji, and NIDN.

```
package Dosen2;

public class InheritanceDemo {
    Run | Debug | Run main | Debug main
    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());
        System.out.println(dosen1.getAllInfo());
    }
}
```

PROBLEMS 2 OUTPUT DEBUG CONSOLE TERMINAL PORTS DEVDB

- PS D:\TUGAS\SEMESTER 3\PBO\PBO\_SAFRIZ\_THEVIGILANTE\PBO Jobsheett 6> & 'C:\Program Files\Java\jdk-17\bin\java.exe' '-er\workspaceStorage\bc2fdab5133c60e4994da4853e1e0bf4\redhat.java\jdt\_ws\PBO Jobsheett 6\_fc1a2906\bin' 'Dosen2.Inheritit

Objek dari class Pegawai dibuat  
Objek dari class Dosen dibuat  
NIP : 34329837  
Nama : Yansy Ayuningtyas  
Gaji : 3000000.0

NIP: 34329837  
Nama: Yansy Ayuningtyas  
Gaji: 3000000.0  
NIDN: 1989432439

- PS D:\TUGAS\SEMESTER 3\PBO\PBO\_SAFRIZ\_THEVIGILANTE\PBO Jobsheett 6>

```

}

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

```

Setelah This

## B. PERTANYAAN

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

Program 1: Method `getAllInfo()` pada program pertama tidak menggunakan kata kunci `this`, sehingga memanggil atribut langsung dari instance objek tanpa eksplisit. Saat Anda memanggil `dosen1.getAllInfo()`, nama, NIP, gaji, dan NIDN yang diambil adalah dari objek `dosen1`.

java

Copy code

```

info += "NIP: " + nip + "\n";
info += "Nama: " + nama + "\n";
info += "Gaji: " + gaji + "\n";
info += "NIDN: " + nidn + "\n";

```

Program 4: Perbedaannya di sini adalah penggunaan kata kunci `this`. Kata kunci `this` merujuk pada atribut dari objek saat ini (misalnya, `dosen1`). Hasil yang ditampilkan tetap sama karena `this.nip`, `this.nama`, `this.gaji`, dan `this.nidn` mengambil nilai dari objek yang sama (`dosen1`), tetapi dengan lebih jelas menunjuk pada atribut objek yang sedang diproses.

java

Copy code

```

info += "NIP: " + this.nip + "\n";
info += "Nama: " + this.nama + "\n";
info += "Gaji: " + this.gaji + "\n";
info += "NIDN: " + this.nidn + "\n";

```

Program 8: Jika di program ke-8 Anda hanya menambahkan string statis dalam `info`, hasilnya tidak akan menampilkan nilai dari objek `dosen1`, melainkan hanya teks kosong atau simbol yang tidak mengandung informasi dari objek tersebut.

java

Copy code

```
info += "#NIP: \n";
```

```
info += "Nama: \n";
```

```
info += "Gaji: \n";
```

```
info += "NIDN: \n";
```

Pada program 1 dan 4, nama, NIP, dan gaji akan ditampilkan karena atribut-atribut tersebut diambil dari objek (dosen1).

Pada program 8, nama, NIP, dan gaji tidak akan ditampilkan karena Anda hanya memasukkan simbol atau teks statis tanpa mengambil atribut dari objek.

2. Mengapa error terjadi pada program no 6?

Pada program no 6, Anda mencoba menggunakan super untuk mengakses atribut. Dalam konteks pewarisan (inheritance), super digunakan untuk mengakses atribut atau method dari kelas induk. Jika kelas Dosen tidak mewarisi kelas lain yang memiliki atribut nip, nama, gaji, dan nidn, maka akan terjadi error karena atribut tersebut tidak ditemukan di kelas induk.

Jadi, error terjadi karena super.nip, super.nama, super.gaji, dan super.nidn tidak valid jika Dosen tidak memiliki kelas induk yang mendefinisikan atribut-atribut tersebut.

## 7. PERCOBAAN 5 (super & overriding)

### A. TAHAPAN PERCOBAAN

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

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

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

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

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

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

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;
    }
}
```

## B. PERTANYAAN

1. Apakah ada perbedaan method `getInfo()` yang diakses pada langkah 1, 2, dan 3?

- **Langkah 1 dan 2:** `getInfo()` dipanggil dari class `Dosen` karena method tersebut didefinisikan dalam class yang sama. Perbedaannya hanya terletak pada cara pemanggilan: di Langkah 1 menggunakan pemanggilan langsung (`getInfo()`), sedangkan di Langkah 2 menggunakan `this.getInfo()`. Keduanya memanggil method yang sama dalam class `Dosen`.

- **Langkah 3:** `super.getInfo()` dipanggil untuk mengambil implementasi `getInfo()` dari superclass (`Pegawai`). Ini berarti, jika method `getInfo()` di-override dalam class `Dosen`, pemanggilan `super.getInfo()` akan mengambil versi `getInfo()` dari `Pegawai`, bukan dari `Dosen`.

2. Apakah ada perbedaan method `super.getInfo()` dan `this.getInfo()` yang dipanggil dalam method `getAllInfo()` pada langkah 4? Jelaskan!

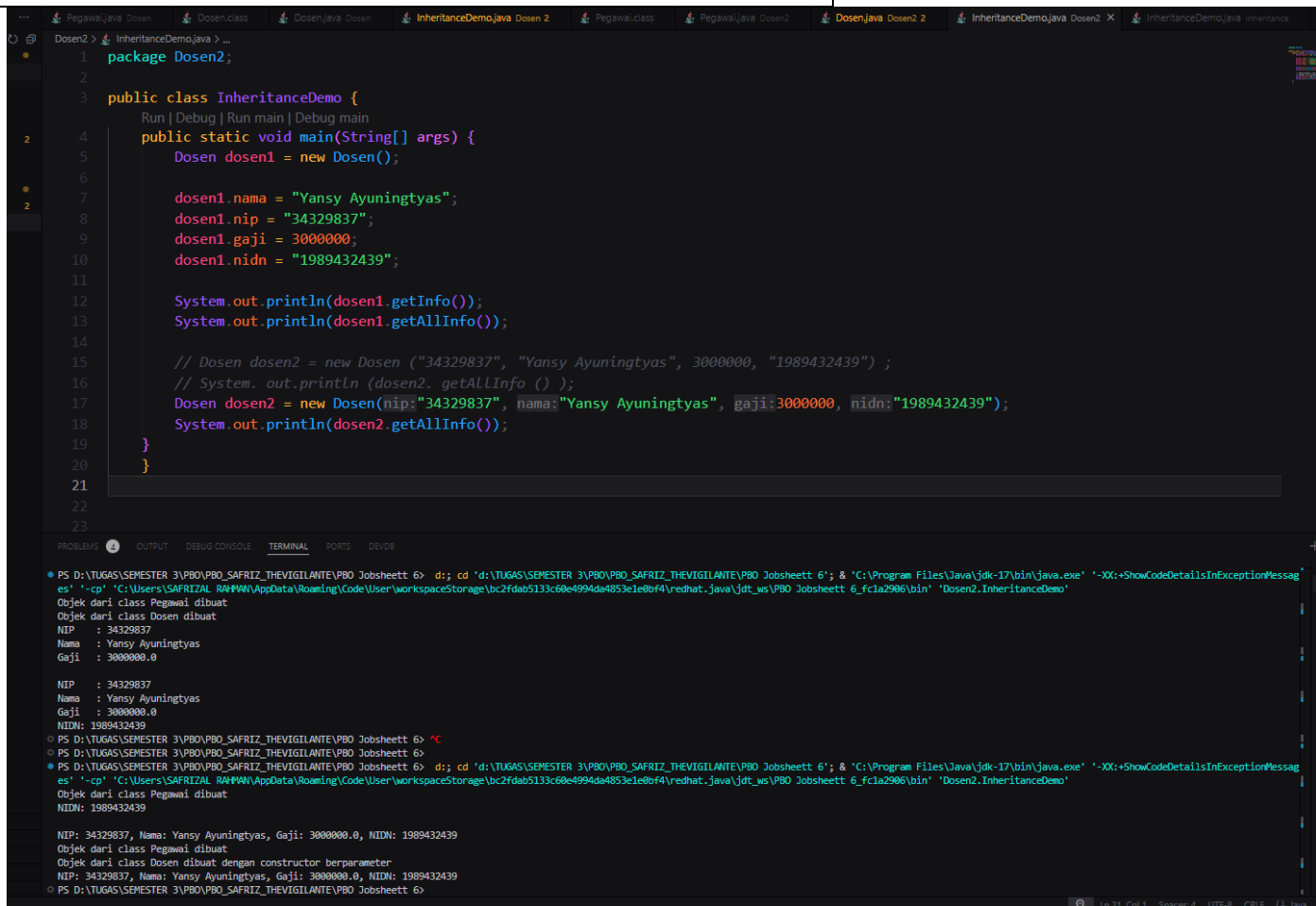
- **`super.getInfo()`:** Memanggil method `getInfo()` dari superclass (`Pegawai`). Jika `Dosen` meng-override method `getInfo()`, maka `super.getInfo()` tetap akan mengambil implementasi dari superclass (`Pegawai`), bukan dari `Dosen`.

- **`this.getInfo()`:** Memanggil method `getInfo()` dari class `Dosen`. Jika `getInfo()` di-override dalam class `Dosen`, maka `this.getInfo()` akan menggunakan versi yang di-override tersebut.

3. Pada method apakah terjadi overriding? Jelaskan!

Overriding terjadi pada method `getInfo()`. Overriding adalah ketika subclass (`Dosen`) mendefinisikan kembali method yang sudah ada di superclass (`Pegawai`) dengan cara yang sama (signature yang sama). Dalam hal ini, method `getInfo()` didefinisikan di `Pegawai`, dan kemudian di-override di class `Dosen` dengan implementasi yang berbeda. Overriding memungkinkan subclass menyediakan implementasi spesifik dari method yang telah didefinisikan oleh superclass.

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



The screenshot shows an IDE with the following code in `InheritanceDemo.java`:

```
package Dosen2;

public class InheritanceDemo {
    Run | Debug | Run main | Debug main
    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());
        System.out.println(dosen1.getAllInfo());

        // Dosen dosen2 = new Dosen ("34329837", "Yansy Ayuningtyas", 3000000, "1989432439") ;
        // System.out.println (dosen2. getAllInfo () );
        Dosen dosen2 = new Dosen(nip:"34329837", nama:"Yansy Ayuningtyas", gaji:3000000, nidn:"1989432439");
        System.out.println(dosen2.getAllInfo());
    }
}
```

The terminal output shows the execution results:

```
PS D:\TUGAS\SEMESTER 3\PRO\PRO_SAFRIZ_THEVIGILANTE\PRO Jobsheet 6> d; cd 'd:\TUGAS\SEMESTER 3\PRO\PRO_SAFRIZ_THEVIGILANTE\PRO Jobsheet 6'; & 'C:\Program Files\Java\jdk-17\bin\java.exe' '-XX:+ShowCodeDetailsInExceptionMessages' '-cp' 'C:\Users\SAFRIZAL_RAHMAN\AppData\Roaming\Code\User\workspaceStorage\bc2fdab5133c60e4994da4853e1e0bf4\redhat-java\jdt_ws\PRO Jobsheet 6_fc1a2906\bin' 'Dosen2.InheritanceDemo'
Objek dari class Pegawai dibuat
Objek dari class Dosen dibuat
NIP : 34329837
Nama : Yansy Ayuningtyas
Gaji : 3000000.0

NIP : 34329837
Nama : Yansy Ayuningtyas
Gaji : 3000000.0
NIDN: 1989432439
PS D:\TUGAS\SEMESTER 3\PRO\PRO_SAFRIZ_THEVIGILANTE\PRO Jobsheet 6> ^C
PS D:\TUGAS\SEMESTER 3\PRO\PRO_SAFRIZ_THEVIGILANTE\PRO Jobsheet 6> d; cd 'd:\TUGAS\SEMESTER 3\PRO\PRO_SAFRIZ_THEVIGILANTE\PRO Jobsheet 6'; & 'C:\Program Files\Java\jdk-17\bin\java.exe' '-XX:+ShowCodeDetailsInExceptionMessages' '-cp' 'C:\Users\SAFRIZAL_RAHMAN\AppData\Roaming\Code\User\workspaceStorage\bc2fdab5133c60e4994da4853e1e0bf4\redhat-java\jdt_ws\PRO Jobsheet 6_fc1a2906\bin' 'Dosen2.InheritanceDemo'
Objek dari class Pegawai dibuat
Objek dari class Dosen dibuat dengan constructor berparameter
NIP: 34329837, Nama: Yansy Ayuningtyas, Gaji: 3000000.0, NIDN: 1989432439
PS D:\TUGAS\SEMESTER 3\PRO\PRO_SAFRIZ_THEVIGILANTE\PRO Jobsheet 6>
```

## 8. PERCOBAAN 6 (overloading)

### A. TAHAPAN PERCOBAAN

1. 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");
}
```

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

```
package Dosen2;

public class InheritanceDemo {
    Run | Debug | Run main | Debug main
    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());
        System.out.println(dosen1.getAllInfo());

        // Dosen dosen2 = new Dosen ("34329837", "Yansy Ayuningtyas", 3000000, "1989432439") ;
        // System.out.println(dosen2.getAllInfo());
        Dosen dosen2 = new Dosen(nip:"34329837", nama:"Yansy Ayuningtyas", gaji:3000000, nidn:"1989432439");
        System.out.println(dosen2.getAllInfo());
    }
}

PS D:\TUGAS\SEMESTER 3\PROJ\PROJ_SAFRIZ_THEVIGILANTE\PROJ Jobsheet 6> d; cd 'd:\TUGAS\SEMESTER 3\PROJ\PROJ_SAFRIZ_THEVIGILANTE\PROJ Jobsheet 6'; & 'C:\Program Files\Java\jdk-17\bin\java.exe' '-XX:+ShowCodeDetailsInExceptionMessages' '-cp' 'C:\Users\SAFRIZAL_RAHMAN\AppData\Roaming\Code\User\workspaceStorage\bc2fdab5133c60e4994da4853e1e0bf4\redhat.java\jdk_ws\PROJ Jobsheet 6_fc1a2906\bin' 'Dosen2.InheritanceDemo'
Objek dari class Pegawai dibuat
NIP : 34329837
Nama : Yansy Ayuningtyas
Gaji : 3000000.0

NIP : 34329837
Nama : Yansy Ayuningtyas
Gaji : 3000000.0
NIDN: 1989432439
PS D:\TUGAS\SEMESTER 3\PROJ\PROJ_SAFRIZ_THEVIGILANTE\PROJ Jobsheet 6> ^C
PS D:\TUGAS\SEMESTER 3\PROJ\PROJ_SAFRIZ_THEVIGILANTE\PROJ Jobsheet 6>
PS D:\TUGAS\SEMESTER 3\PROJ\PROJ_SAFRIZ_THEVIGILANTE\PROJ Jobsheet 6> d; cd 'd:\TUGAS\SEMESTER 3\PROJ\PROJ_SAFRIZ_THEVIGILANTE\PROJ Jobsheet 6'; & 'C:\Program Files\Java\jdk-17\bin\java.exe' '-XX:+ShowCodeDetailsInExceptionMessages' '-cp' 'C:\Users\SAFRIZAL_RAHMAN\AppData\Roaming\Code\User\workspaceStorage\bc2fdab5133c60e4994da4853e1e0bf4\redhat.java\jdk_ws\PROJ Jobsheet 6_fc1a2906\bin' 'Dosen2.InheritanceDemo'
Objek dari class Pegawai dibuat
NIP: 1989432439

NIP: 34329837, Nama: Yansy Ayuningtyas, Gaji: 3000000.0, NIDN: 1989432439
Objek dari class Pegawai dibuat
Objek dari class Dosen dibuat dengan constructor berparameter
NIP: 34329837, Nama: Yansy Ayuningtyas, Gaji: 3000000.0, NIDN: 1989432439
PS D:\TUGAS\SEMESTER 3\PROJ\PROJ_SAFRIZ_THEVIGILANTE\PROJ Jobsheet 6>
```

## B. PERTANYAAN

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

The values of nip, nama, gaji, and nidn are displayed because they are initialized in the parameterized constructor when the Dosen object (dosen2) is created. The constructor assigns these values to the object's fields, which can then be accessed through the getAllInfo() method.

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

No, they do not have the same signature. The signature of a constructor (or method) includes its name and the number and types of its parameters. The no-argument constructor has no parameters, while the parameterized constructor has four parameters (String, String, double, String). Thus, they have different signatures.

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

This concept is known as **method overloading**. In Java, a class can have multiple methods (or constructors) with the same name as long as they have different signatures (i.e., a different number of parameters or different parameter types). This allows for flexibility and improved readability in your code.

---

## 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;  
}
```

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;  
}
```

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;  
}
```

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;
    }
}

```

5. 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);
}

```

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;
}

```

## B. PERTANYAAN

1. Apakah terdapat perbedaan hasil pada langkah 1 dan 2? Jelaskan!

In step 1, the attributes are initialized using `this`, directly referencing the current instance. In step 2, `super` is used, which only works if the attributes are accessible and inherited. If they are not, an error will occur.

2. Apakah terdapat perbedaan hasil pada langkah 2 dan 3? Jelaskan!

---

### **Difference between Steps 2 and 3:**

In step 3, `super()` is explicitly called before setting the attributes. If `super()` initializes properly and the superclass has those attributes, it should work as expected. If it's not, it will lead to an error.

3. Mengapa terjadi error pada langkah 4?

If the default constructor from the `Pegawai` class is commented out or deleted, you can't create an instance of `Pegawai` without parameters. The code tries to instantiate `Pegawai` with parameters, leading to a compile-time error if no suitable constructor exists.

4. Apa perbedaan `super()` yang dipanggil pada langkah 3 dan 6?

In step 3, `super()` is called but assigns values afterward. In step 6, the `super()` call initializes inherited attributes properly before the subclass initializes its own attributes, which is the correct practice.

5. Mengapa terjadi error pada langkah 5?

In step 5, the error occurs because `this.nidn = nidn;` is written before calling `super()`. In Java, `super()` must be the first statement in the constructor.

```

1 package Dosen2;
2
3 public class InheritanceDemo {
4     Run | Debug | Run main | Debug main
5     public static void main(String[] args) {
6         Dosen dosen1 = new Dosen();
7
8         dosen1.nama = "Yansy Ayuningtyas";
9         dosen1.nip = "34329837";
10        dosen1.gaji = 3000000;
11        dosen1.nidn = "1989432439";
12
13        System.out.println(dosen1.getInfo());
14        System.out.println(dosen1.getAllInfo());
15
16        // Dosen dosen2 = new Dosen ("34329837", "Yansy Ayuningtyas", 3000000, "1989432439") ;
17        // System.out.println (dosen2. getAllInfo () );
18        Dosen dosen2 = new Dosen(nip:"34329837", nama:"Yansy Ayuningtyas", gaji:3000000, nidn:"1989432439");
19        System.out.println(dosen2.getAllInfo());
20    }
21 }
22
23

```

PROBLEMS OUTPUT DEBUG CONSOLE **TERMINAL** PORTS DEVS

```

Nama : Yansy Ayuningtyas
Gaji : 3000000.0
NIDN: 1989432439
PS D:\TUGAS\SEMESTER 3\PBO\PBO_SAFRIZ_THEVIGILANTE\PBO Jobsheett 6> ^C
PS D:\TUGAS\SEMESTER 3\PBO\PBO_SAFRIZ_THEVIGILANTE\PBO Jobsheett 6>
PS D:\TUGAS\SEMESTER 3\PBO\PBO_SAFRIZ_THEVIGILANTE\PBO Jobsheett 6> d:; cd 'd:\TUGAS\SEMESTER 3\PBO\PBO_SAFRIZ_THEVIGILANTE\PBO Jobsheett 6'; & 'C:\Program Files\Java\jdk-17\bin\java.exe' '-cp' 'C:\Users\SAFRIZAL RAHMAN\AppData\Roaming\Code\User\workspaceStorage\bc2fdab5133c60e4994da4853e1e0bf4\redhat.java\jdt_ws\PBO Jobsheett 6_fc1a2986\bin' 'Dosen2.InheritanceDemo'
Objek dari class Pegawai dibuat
NIDN: 1989432439

NIP: 34329837, Nama: Yansy Ayuningtyas, Gaji: 3000000.0, NIDN: 1989432439
Objek dari class Pegawai dibuat
Objek dari class Dosen dibuat dengan constructor berparameter
NIP: 34329837, Nama: Yansy Ayuningtyas, Gaji: 3000000.0, NIDN: 1989432439
PS D:\TUGAS\SEMESTER 3\PBO\PBO_SAFRIZ_THEVIGILANTE\PBO Jobsheett 6> ^C
PS D:\TUGAS\SEMESTER 3\PBO\PBO_SAFRIZ_THEVIGILANTE\PBO Jobsheett 6>
PS D:\TUGAS\SEMESTER 3\PBO\PBO_SAFRIZ_THEVIGILANTE\PBO Jobsheett 6> d:; cd 'd:\TUGAS\SEMESTER 3\PBO\PBO_SAFRIZ_THEVIGILANTE\PBO Jobsheett 6'; & 'C:\Program Files\Java\jdk-17\bin\java.exe' '-cp' 'C:\Users\SAFRIZAL RAHMAN\AppData\Roaming\Code\User\workspaceStorage\bc2fdab5133c60e4994da4853e1e0bf4\redhat.java\jdt_ws\PBO Jobsheett 6_fc1a2986\bin' 'Dosen2.InheritanceDemo'
Objek dari class Pegawai dibuat
NIDN: 1989432439

NIP: 34329837, Nama: Yansy Ayuningtyas, Gaji: 3000000.0, NIDN: 1989432439
NIP: 34329837, Nama: Yansy Ayuningtyas, Gaji: 3000000.0, NIDN: 1989432439
PS D:\TUGAS\SEMESTER 3\PBO\PBO_SAFRIZ_THEVIGILANTE\PBO Jobsheett 6>

```

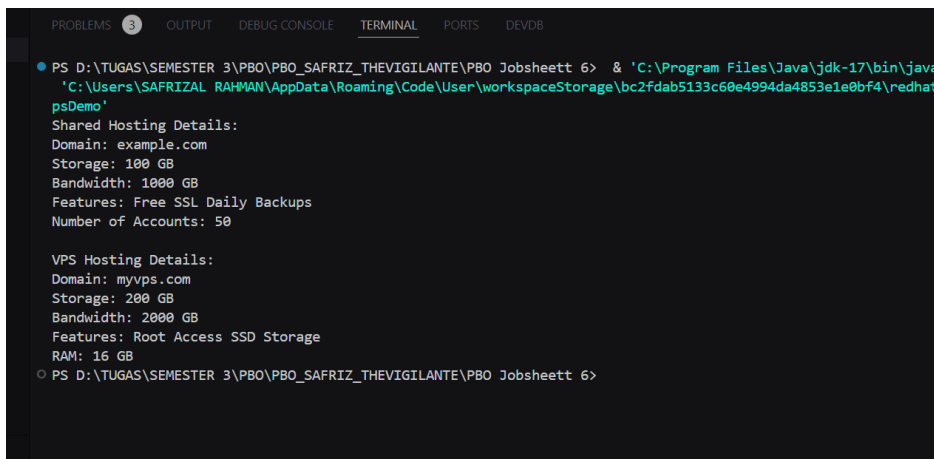
## 10. TUGAS

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

---

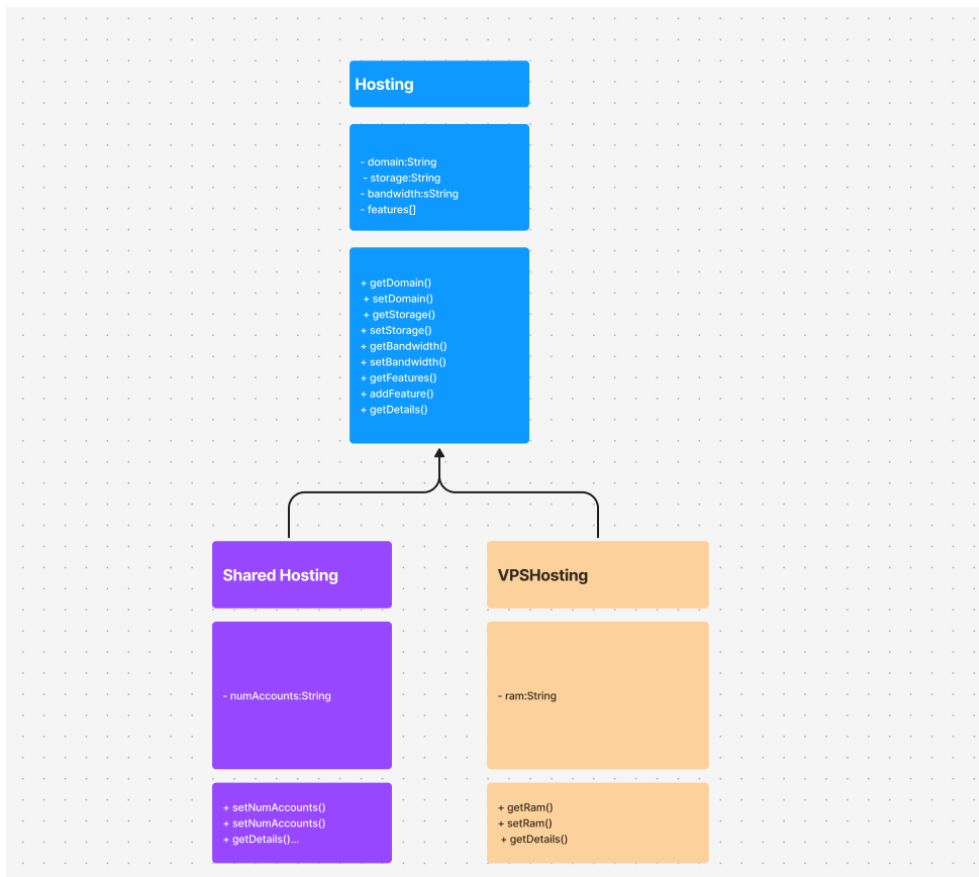
parameter dan constructor berparameter pada masing-masing class. Panggil constructor `super()` berparameter untuk membuat object dari parent class pada constructor child class.

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.



```
PROBLEMS 3 OUTPUT DEBUG CONSOLE TERMINAL PORTS DEVDB
PS D:\TUGAS\SEMESTER 3\PBO\PBO_SAFRIZ_THEVIGILANTE\PBO Jobsheett 6> & 'C:\Program Files\Java\jdk-17\bin\java
'C:\Users\SAFRIZAL RAHMAN\AppData\Roaming\Code\User\workspaceStorage\bc2fdab5133c60e4994da4853e1e0bf4\redhat
psDemo'
Shared Hosting Details:
Domain: example.com
Storage: 100 GB
Bandwidth: 1000 GB
Features: Free SSL Daily Backups
Number of Accounts: 50

VPS Hosting Details:
Domain: myvps.com
Storage: 200 GB
Bandwidth: 2000 GB
Features: Root Access SSD Storage
RAM: 16 GB
PS D:\TUGAS\SEMESTER 3\PBO\PBO_SAFRIZ_THEVIGILANTE\PBO Jobsheett 6>
```



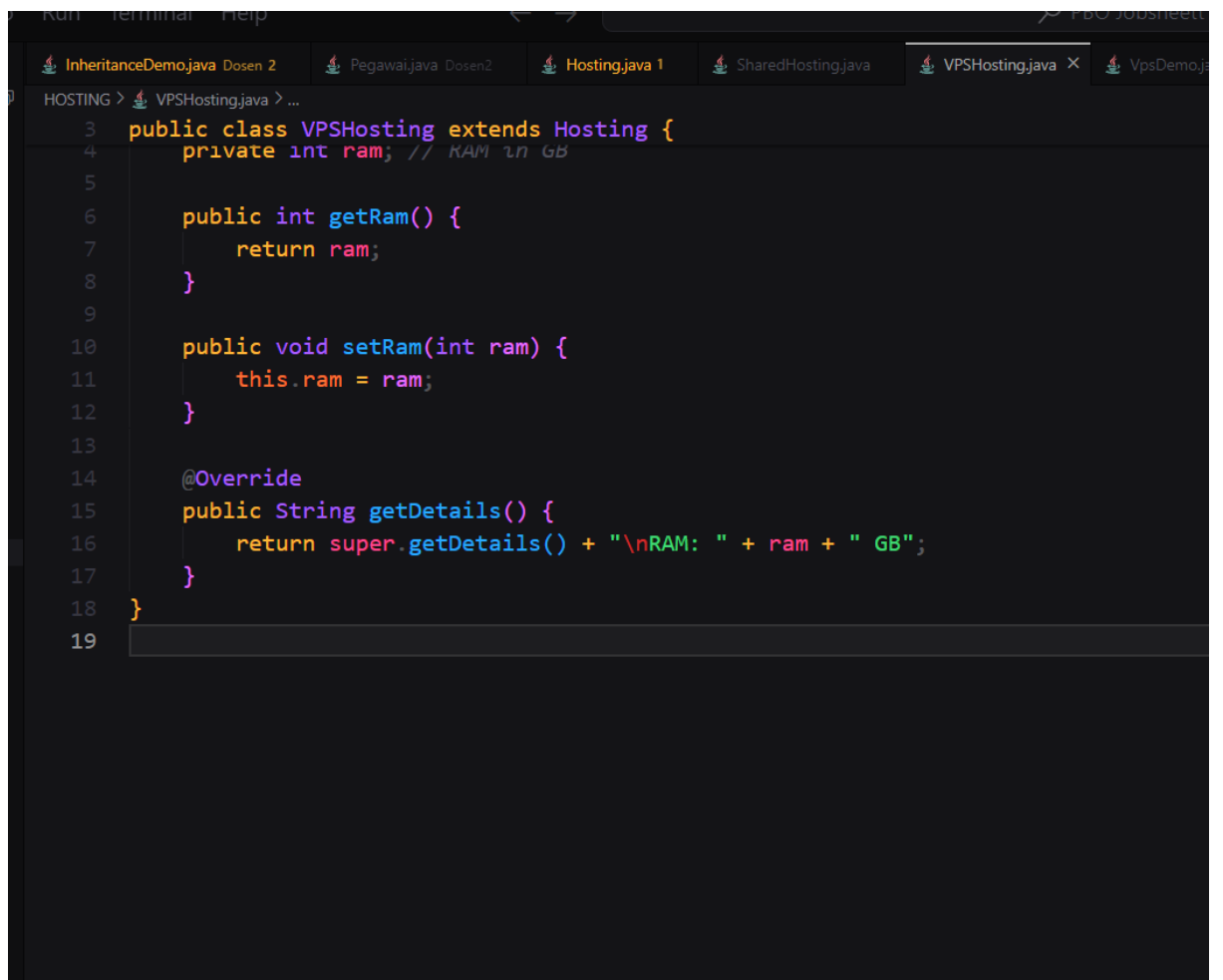
```
HOSTING > VpsDemo.java > Language Support for Java(TM) by Red Hat > VpsDemo > main(String[])
1 package HOSTING;
2
3 public class VpsDemo {
4     Run | Debug | Run main | Debug main
5     public static void main(String[] args) {
6         // SharedHosting instance
7         SharedHosting shared = new SharedHosting();
8         shared.setDomain(domain:"example.com");
9         shared.setStorage(storage:100);
10        shared.setBandwidth(bandwidth:1000);
11        shared.setNumAccounts(numAccounts:50);
12        shared.addFeature(feature:"Free SSL");
13        shared.addFeature(feature:"Daily Backups");
14        System.out.println(x:"Shared Hosting Details:");
15        System.out.println(shared.getDetails());
16
17        // VPSHosting instance
18        VPSHosting vps = new VPSHosting();
19        vps.setDomain(domain:"myvps.com");
20        vps.setStorage(storage:200);
21        vps.setBandwidth(bandwidth:2000);
22        vps.setRam(ram:16);
23        vps.addFeature(feature:"Root Access");
24        vps.addFeature(feature:"SSD Storage");
25        System.out.println(x:"\nVPS Hosting Details:");
26        System.out.println(vps.getDetails());
27    }
28 }
```

```
InheritanceDemo.java Dosen 2  Pegawai.java Dosen2  Hosting.java 1  SharedHosting.java X  VPSHosting.java  VpsDemo.java  Dosen.java Do

HOSTING > SharedHosting.java > ...
1  package HOSTING;
2
3  public class SharedHosting extends Hosting {
4      private int numAccounts;
5
6      public int getNumAccounts() {
7          return numAccounts;
8      }
9
10     public void setNumAccounts(int numAccounts) {
11         this.numAccounts = numAccounts;
12     }
13
14     @Override
15     public String getDetails() {
16         return super.getDetails() + "\nNumber of Accounts: " + numAccounts;
17     }
18 }
19
```

```
InheritanceDemo.java Dosen 2  Pegawai.java Dosen2  Hosting.java 1  SharedHosting.java  VPSHosting.java  VpsDemo.java x  Dosenj
HOSTING > VpsDemo.java > Language Support for Java(TM) by Red Hat > VpsDemo > main(String[])
1  package HOSTING;
2
3  public class VpsDemo {
    Run | Debug | Run main | Debug main
4      public static void main(String[] args) {
5          // SharedHosting instance
6          SharedHosting shared = new SharedHosting();
7          shared.setDomain(domain:"example.com");
8          shared.setStorage(storage:100);
9          shared.setBandwidth(bandwidth:1000);
10         shared.setNumAccounts(numAccounts:50);
11         shared.addFeature(feature:"Free SSL");
12         shared.addFeature(feature:"Daily Backups");
13         System.out.println(x:"Shared Hosting Details:");
14         System.out.println(shared.getDetails());
15
16         // VPSHosting instance
17         VPSHosting vps = new VPSHosting();
18         vps.setDomain(domain:"myvps.com");
19         vps.setStorage(storage:200);
20         vps.setBandwidth(bandwidth:2000);
21         vps.setRam(ram:16);
22         vps.addFeature(feature:"Root Access");
23         vps.addFeature(feature:"SSD Storage");
24         System.out.println(x:"\nVPS Hosting Details:");
25         System.out.println(vps.getDetails());
26     }
27 }
28
```



A screenshot of an IDE window showing a Java file named VPSHosting.java. The code defines a public class VPSHosting that extends a class named Hosting. It includes a private integer attribute ram, and methods getRam(), setRam(), and getDetails(). The getDetails() method is annotated with @Override and concatenates the output of super.getDetails() with a string representing the RAM value. The IDE interface includes a menu bar (Run, Terminal, Help), a tab bar with several open files, and a breadcrumb trail (HOSTING > VPSHosting.java > ...).

```
3 public class VPSHosting extends Hosting {
4     private int ram; // RAM in GB
5
6     public int getRam() {
7         return ram;
8     }
9
10    public void setRam(int ram) {
11        this.ram = ram;
12    }
13
14    @Override
15    public String getDetails() {
16        return super.getDetails() + "\nRAM: " + ram + " GB";
17    }
18 }
19
```

--- selamat mengerjakan----