# JOBSHEET 6

**INHERITANCE & POLYMORPHISM**

# KOMPETENSI

* 1. Memahami konsep dasar inheritance dan polymorphism
  2. Mampu membuat suatu subclass dari suatu superclass tertentu.
  3. Mampu membuat objek dari suatu subclass dan melakukan pengaksesan terhadap atribut dan method baik yang dimiliki sendiri atau turunan dari superclass nya.
  4. Mampu membuat method overloading
  5. Mampu membuat method overriding

# 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 {

...

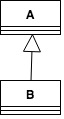
}

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 panambahan atribut dan method khusus yang hanya dimiliki oleh class B.

Terdapat 3 bentuk pewarisan: single inheritance, multilevel inheritance, dan multiple inheritance.

1. Single Inheritance

Single inheritance adalah inheritance dimana suatu subclass hanya mempunyai satu parent class.



Gambar 1. Contoh Single Inheritance

1. Multilevel Inheritance

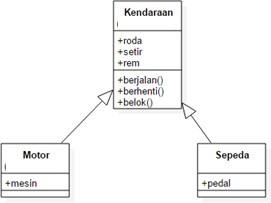
Multilevel inheritance adalah inheritance dengan subclass yang menjadi superclass bagi class yang lain.

Contoh:

Gambar 2. Contoh Multilevel Inheritance

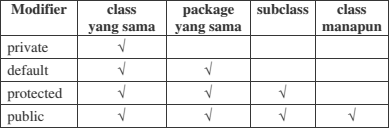
Pada Gambar 2 di atas dapat dilihat bahwa class B merupakan subclass dari class A, namun dia juga merupakan superclass dari class C. Inheritance dengan 2 level ini disebut multilevel inheritance.

Pada class diagram, inheritance digambarkan dengan sebuah garis solid dengan segitiga di ujungnya. Class yang dekat pada segitiga merupakan superclass, sedangkan class yang jauh dari segitiga merupakan subclass. Berikut ini adalah contoh class diagram dengan relasi inheritance:



Gambar 3 Contoh class diagram dalam inheritance

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:



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** dipakai untuk merujuk pada object/class itu sendiri. Sementara itu, kunci **super** dipakai untuk merujuk pada parent object/class. Format penulisannya adalah sebagai berikut:

### super.<namaAtribut>

Mengakses atribut parent

### super.<namaMethod>()

Memanggil method parent

### super()

Memganggil constructor parent, hanya dapat dilakukan pada baris pertama dalam constructor child

### super(parameter1, parameter2,dst)

Memanggil constructor parent class dengan parameter

Saat instansiasi objek dari subclass dilakukan, objek pada superclass juga akan terbentuk. Dengan kata lain, ketika constructor subclass dijalankan, pada “baris pertama” (atau sebelum baris- baris lainnya dalam constructor subclass dieksekusi) constructor superclass akan dijalankan terlebih dahulu.

**Polymorphism** terdiri dari 2 kata, yaitu poly (banyak), morph (bentuk). Konsep polimorfisme pada OOP membolehkan sebuah aksi diimplementasikan secara berbeda. Ada 2 bentuk polimorfisme, yaitu:

### Overloading

* + - Method overloading berarti kondisi dimana ada method dengan nama yang sama, tetapi memiliki method signature yang berbeda.
    - Method signature: jumlah, tipe data dan susunan parameter
    - Method overloading dapat terjadi pada kelas yang sama atau kelas lain yang terkait dalam hierarki pewarisan.
    - Karakteristik overloading: nama method sama, method signature berbeda, return type boleh sama atau berbeda.
    - JVM menentukan method mana yang akan dipanggil pada compile-time  compile-time polymorphism
    - Disebut juga static binding atau early binding

### Overriding

* + - Overriding terjadi ketika child class memiliki method dengan nama dan signature yang sama dengan parent class nya.
    - Karakteristik: terjadi pada child class/kelas turunan, nama method sama, method signature sama.
    - JVM menentukan method mana yang akan dipanggil pada saat run-time  run-time polymorphism
    - Disebut juga dynamic binding atau late binding

# PERCOBAAN 1 (extends)

## 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");

    }

}

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

class Dosen *extends* *Pegawai*{

*public* Dosen(){

        System.*out*.println("Objek dari class Dosen dibuat");

    }

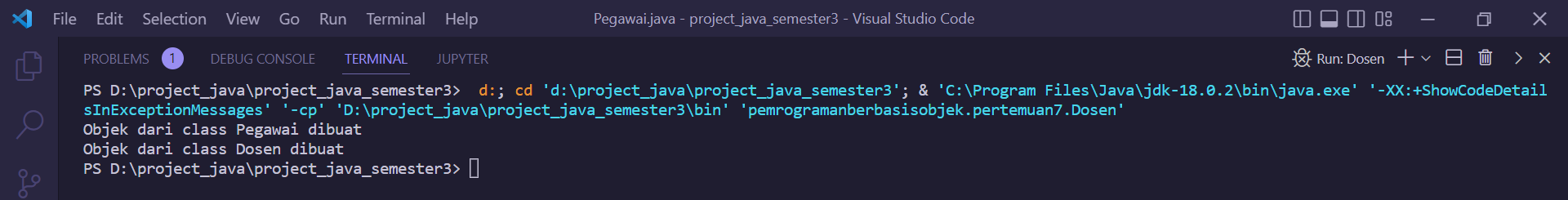
}

1. 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();

    }

1. Run programnya kemudian amati hasilnya.

## B. PERTANYAAN

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

Jawab :

1. Child class : Dosen
2. Parent class : Pegawai
3. Kata kunci apa yang membuat child class dan parent class tersebut memiliki relasi?

Jawab : Menggunakan kata kunci **extends**

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

Jawab : ada 2 constructor yang dieksekusi, constructor class Pegawai yang dieksekusi terlebih dahulu.

# PERCOBAAN 2 (Pewarisan)

## 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\t\t\t\t: " **+** nip **+** "\n";

        info **+=** "Nama\t\t\t\t: " **+** nama **+** "\n";

        info **+=** "Gaji\t\t\t\t: " **+** gaji **+** "\n";

**return** info;

    }

}

* + - 1. Tambahkan pula atribut NIDN pada class Dosen

class Dosen *extends* *Pegawai*{

*public* String nidn;

*public* Dosen(){

        System.*out*.println("Objek dari class Dosen dibuat");

}

* + - 1. 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;

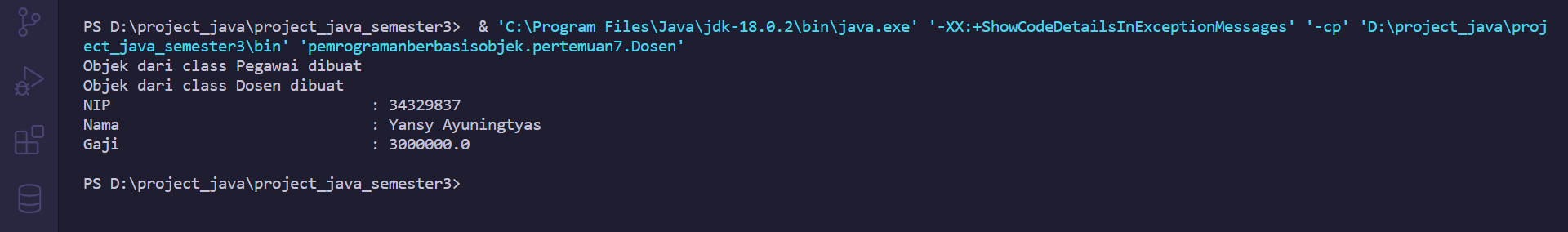
        dosen1.*nidn* **=** "1989432439";

        System.*out*.println(dosen1.getInfo());

    }

}

* + - 1. Run program kemudian amati hasilnya



## PERTANYAAN

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

Jawab : Iya, program berhasil menjalankan

* + - 1. 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?

Jawab : karena terdapat keyword **extends** sebagai deklarasi. keyword extends tersebut memberitahu kompiler Java bahwa kita ingin melakukan **extension/ perluasan** class.

* + - 1. Jika program berhasil dijalankan, mengapa tidak terjadi error pada pemanggilan method getInfo() oleh object dosen1 padahal tidak ada deklarasi method getInfo() pada class Dosen?

Jawab : Karena jika class Dosen meng-extends dari class Pegawai maka akan otomatis atribut dan methodnya akan dideklarasikan (otomatis mewarisi atribut dan method dari parent class).

# PERCOBAAN 3 (Hak akses)

## TAHAPAN PERCOBAAN

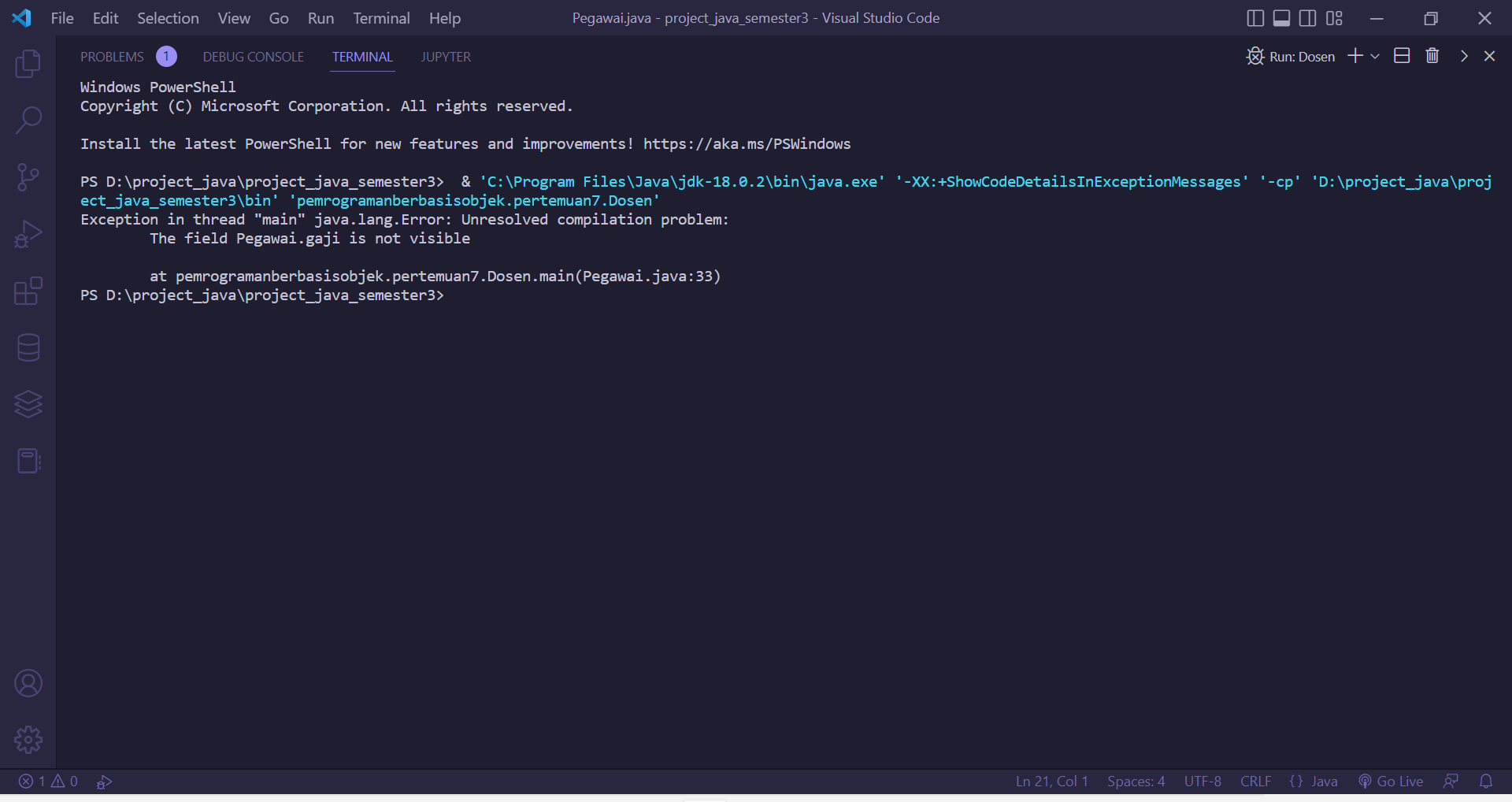
* + 1. Modifikasi access level modifier pada atribut gaji menjadi private pada class Pegawai.java

*public* class Pegawai {

*public* String nip;

*public* String nama;

*private* *double* gaji;

* + 1. Run program kemudian amati hasilnya.
    2. Ubah access level modifier atribut gaji menjadi protected kemudian pindah class Pegawai ke package baru, misalnya “testpackage”.

**package** pemrogramanberbasisobjek.pertemuan7.testpackage;

*public* class Pegawai {

*public* String nip;

*public* String nama;

*protected* *double* gaji;

* + 1. Import class Pegawai dari testpackage pada class Dosen.

**package** pemrogramanberbasisobjek.pertemuan7.inheritance;

*import* pemrogramanberbasisobjek.pertemuan7.testpackage.Pegawai;

* + 1. Akses atribut gaji pada class Dosen dengan coba mencetak atribut gaji pada constructor Dosen

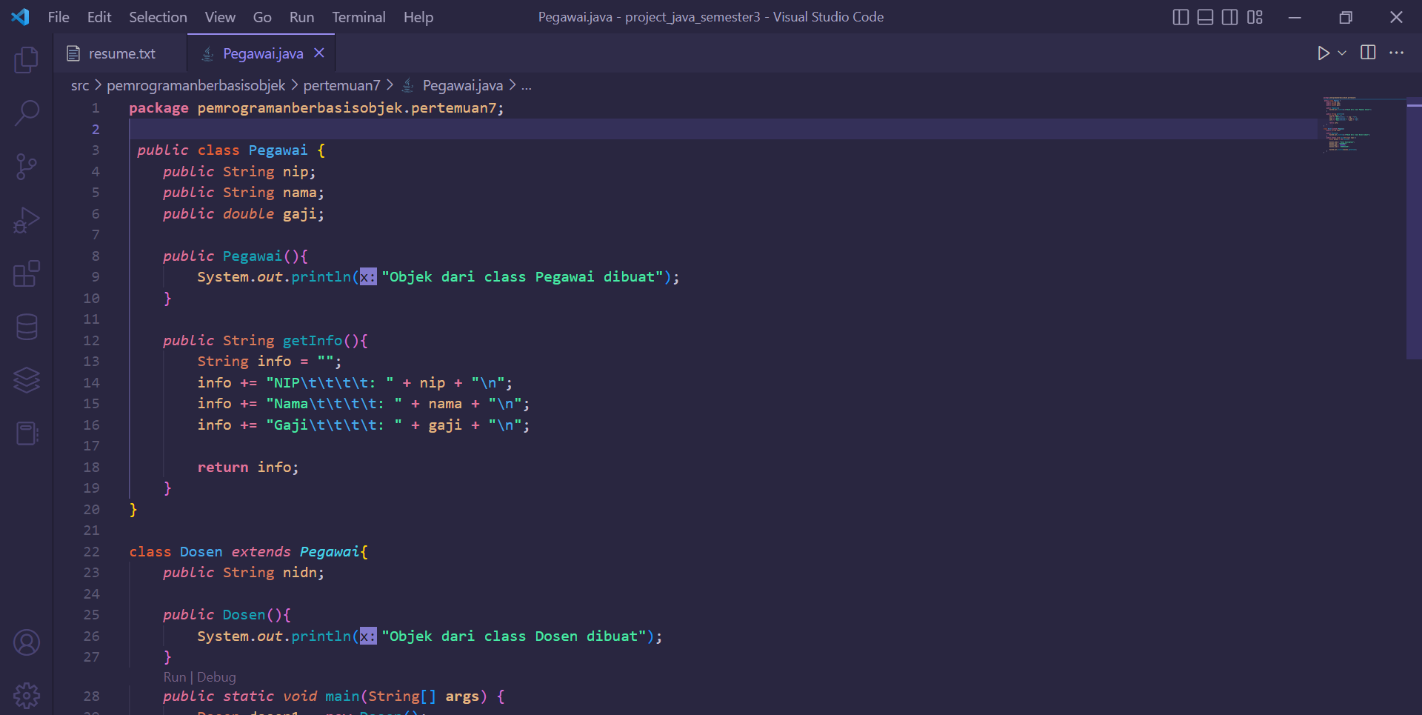
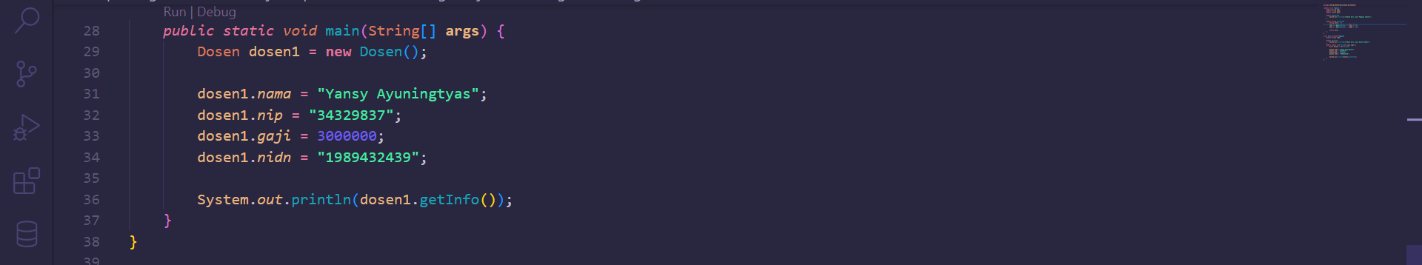
*public* Dosen(){

        System.*out*.println(gaji);

        System.*out*.println("Objek dari class Dosen dibuat");

}

* + 1. Ubah kembali access level modifier menjadi public dan kembalikan class Pegawai ke package semula.



## 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?

Jawab : Karena pada atribut gaji acces modifiernya diset sebagai private, sehingga pada saat diprivate hanya dapat mengakses pada class yang sama, meskipun gaji merupakan atribut Pegawai yang merupakan parent class dari Dosen.

* + 1. Pada langkah 5, setelah class Pegawai berpindah ke package yang berbeda, class Dosen masih dapat mengakses atribut gaji. Mengapa?

Jawab : Karena pada atribut gaji acces modifiernya diubah menjadi protected, jadi masih dapat mengakses class Pegawai setelah berpindah ke package yang berbeda

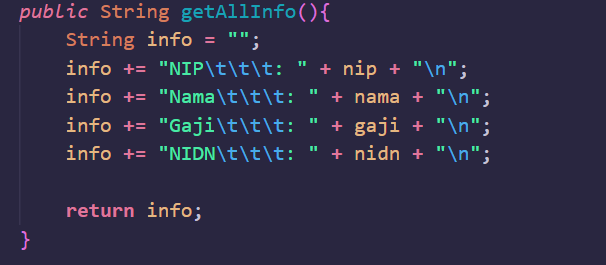
* + 1. Berdasarkan percobaan tersebut, bagaimana menentukan atribut dan method yang akan diwariskan oleh parent class ke child class?

Jawab : Dengan cara pada semua atribut dan method acces modifiernya tidak diset private. Jadi acces modifier boleh public, protected boleh diluar package asal merupakan subclass (pewarisan terhadap class nya), dan no modifier asal di dalam package yang sama.

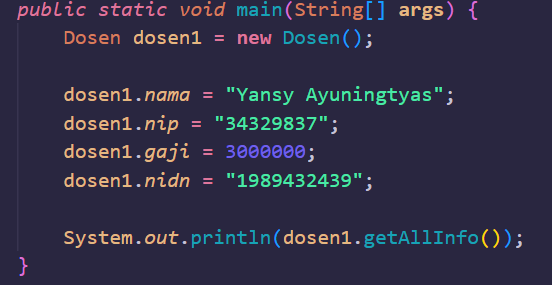
# PERCOBAAN 4 (Super - atribut)

## TAHAPAN PERCOBAAN

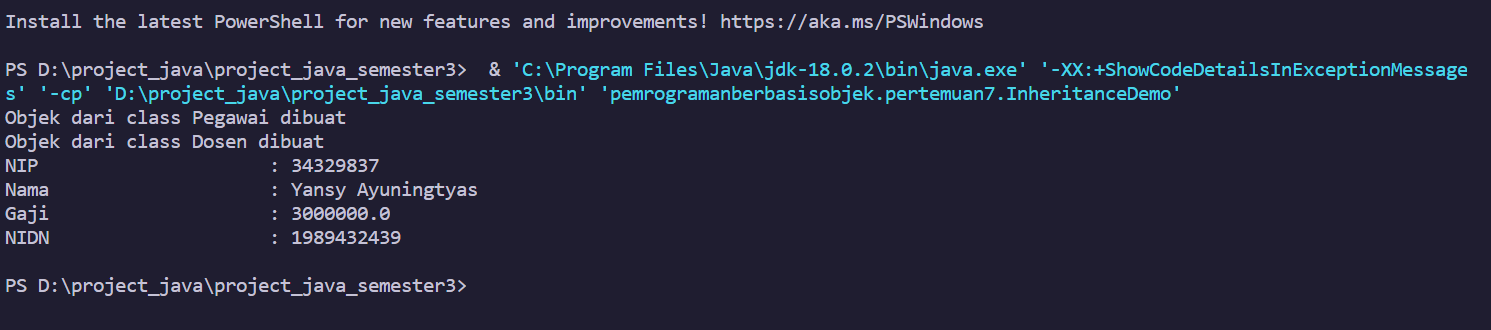
* + 1. Butlah method getAllInfo() pada class Dosen



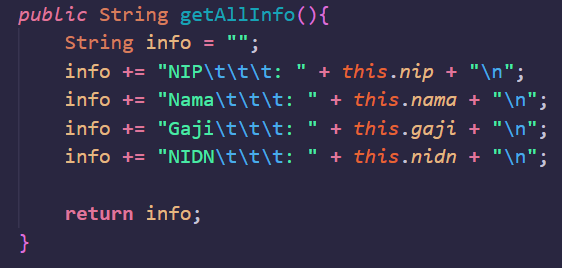
* + 1. Lakukan pemanggilan method getAllInfo() oleh object dosen1 pada class InheritanceDemo.java



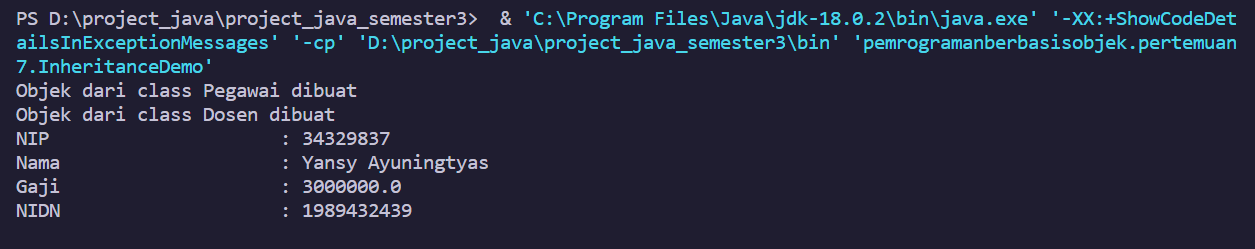
* + 1. Run program kemudian amati hasilnya



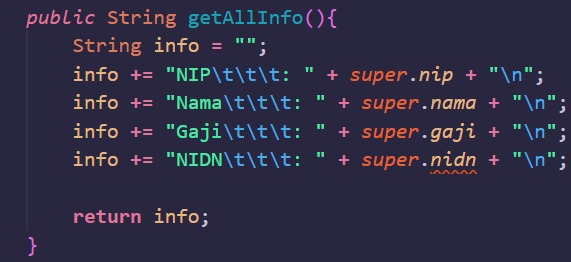
* + 1. Lakukan modifikasi method getAllInfo() pada class Dosen



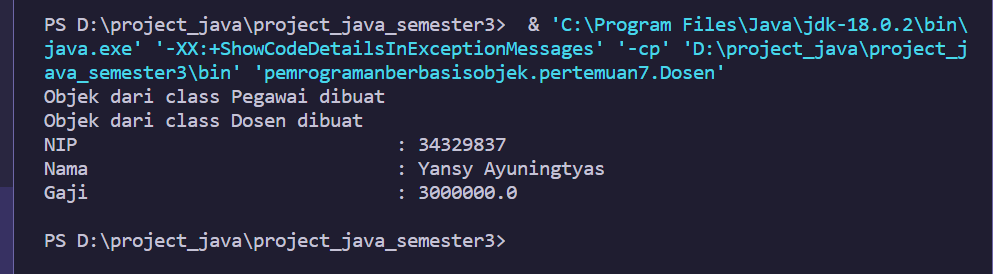
* + 1. Run program kemudian bandingkan hasilnya dengan langkah no 2.



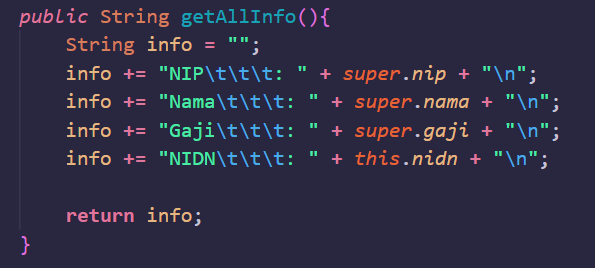
* + 1. Lakukan modifikasi method getAllInfo() pada class Dosen kembali



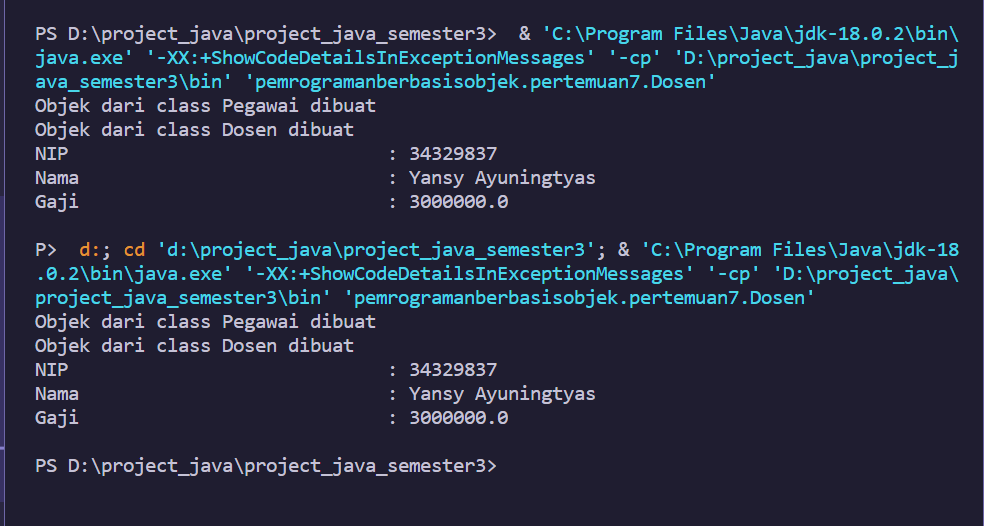
* + 1. Run program kemudian bandingkan hasilnya dengan progam pada no 1 dan no 4.

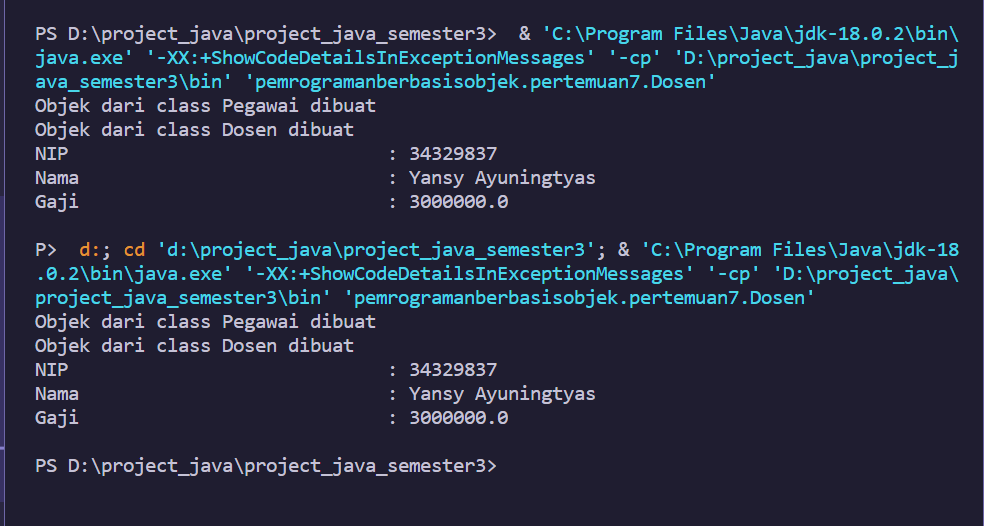


* + 1. Lakukan modifikasi method getAllInfo() pada class Dosen kembali



* + 1. Run program kemudian bandingkan hasilnya dengan progam pada no 2 dan no 4.





## PERTANYAAN

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

Jawab : iya, ada

* Pada program 1 dan 4 nip, nama, gaji, nidn dapat ditampilkan dengan lengkap
* Pada program 8 nidn tidak ditampilkan

Terjadi perbedaan pada program 8 karena, pada program 8 atribut karena Ketika memakai keyword this pada nidn, maka nidn tidak ditampilkan, keyword this akan merujuk class child. Sedangkan pada class parent tidak ada atribut nidn, jadi nidn tidak akan ditampilkan.

Lalu pada program 1 dan program 4 akan memiliki kesamaan karena pada program 1 ketika tidak memakai keyword otomatis atribut dan method akan diwariskan kepada class child dan pada program 4 atribut nip, nama, dan gaji memakai keyword super maka akan langsung merujuk pada atribut di class parent dan this dipakai pada nidn jadi bisa ditampilkan.

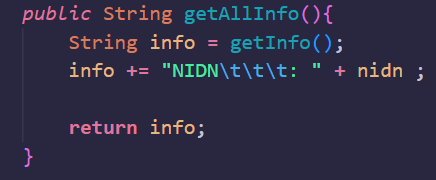
* + 1. Mengapa error terjadi pada program no 6?

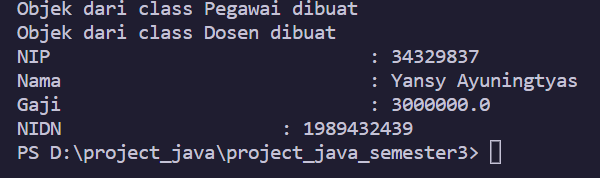
Jawab : Karena keyword **super** akan langsung merujuk pada class parent, sedangkan class parent tidak ada atribut nidn.

# PERCOBAAN 5 (super & overriding )

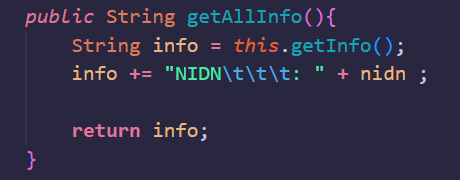
## TAHAPAN PERCOBAAN

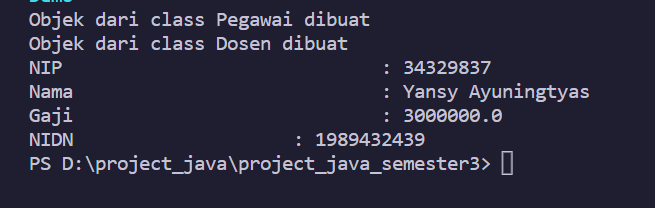
* + 1. Lakukan modifikasi kembali pada method getAllInfo(). Run program kemudian amati hasilnya



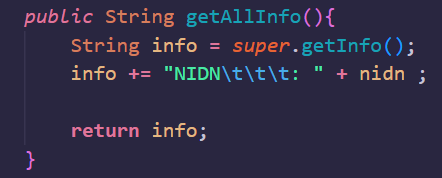


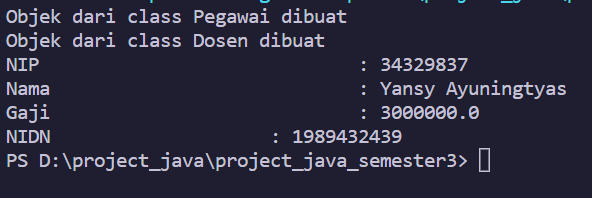
* + 1. Lakukan modifikasi kembali pada method getAllInfo(). Run program kemudian amati hasilnya



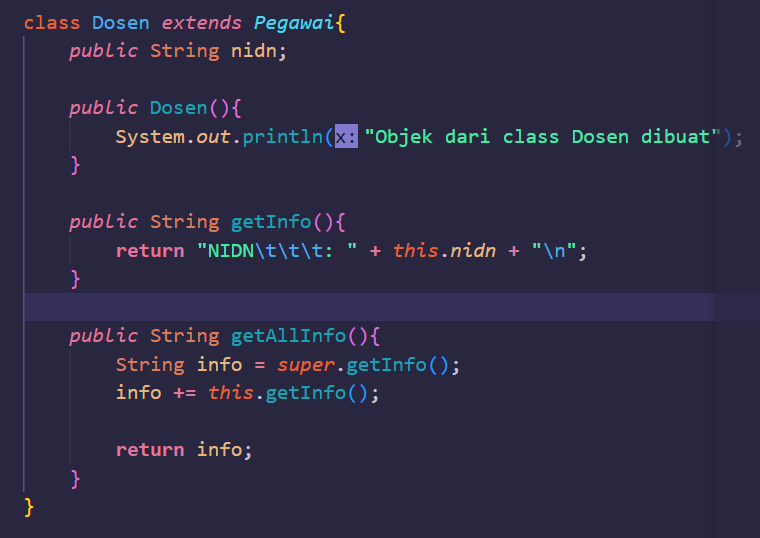


* + 1. Lakukan modifikasi kembali pada method getAllInfo(). Run program kemudian amati hasilnya





* + 1. Tambahkan method getInfo() pada class Dosen dan modifikasi method getAllInfo() sebagai berikut



## PERTANYAAN

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

Jawab : tidak ada, semua menampilkan hasil yang sama.

* Langkah 1 pada subclass Dosen method getAllInfo() sintaks String info = getInfo(); , sintaks tersebut berarti menginisiasi variable info dengan method yang ada di parent class dan pada parent class ada method getInfo() juga maka akan langsung diwariskan pada subclass
* Langkah 2

String info = this.getInfo(); sintaks tersebut akan merujuk pada method yang ada di subclass lalu sintaks info += "NIDN\t\t\t: " + nidn + "\n"; merujuk pada atribut di subclass sehingga hasilnya akan sama pada Langkah 1

* Langkah 3 sintaks String info = super.getInfo(); maka yang akan diakses akan langsung merujuk di parent class lalu di bawahnya ditambah sintaks sintaks info += "NIDN\t\t\t: " + nidn + "\n"; maka hasilnya akan sama pada langkahnya sebelumnya.
  + 1. Apakah ada perbedaan method super.getInfo() dan this.getInfo() yang dipanggil dalam method getAllInfo() pada langkah 4? Jelaskan!

Jawab :

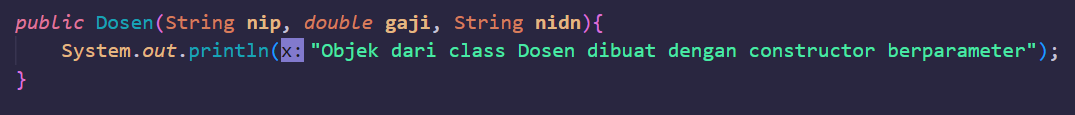
* method super.getInfo() terdapat keyword **super** maka akan merujuk pada method di parent class, yang dipanggil akan langsung menuju pada parent class
* method this.getInfo() keyword **this** , merujuknya hanya di method subclass, maka yang dipanggil hanya tertuju pada subclass
  + 1. Pada method apakah terjadi overriding? Jelaskan!

Jawab : Iya, terjadi overriding. Pada kedua class terdapat method getInfo() yang memiliki signature yang sama sehingga method getInfo() di kelas subclass akan meng-override method getInfo() di parent classnya.

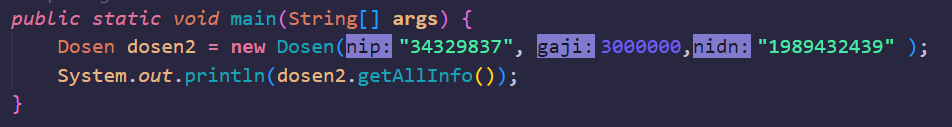
# PERCOBAAN 6 (overloading)

## TAHAPAN PERCOBAAN

* + 1. Tambahkan constructor baru untuk class Dosen sebagai berikut



* + 1. Modifikasi class InheritanceDemo untuk menginstansiasi object baru dengan nama dosen2 dengan constructor yang berparameter. Run program kemudian amati hasilnya.



## PERTANYAAN

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

Jawab : hasil NIP, nama, NIDN akan bernilai null, dan untuk gaji bernilai 0 karena tidak ada perintah (keyword) manakah yang harus dirujuk/diakses terhadapa class parent maupun subclass

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

Jawab : Iya beda, karena constructor tanpa parameter dengan berparameter jelas memiliki signature yang beda dari susunan, tipe data, dan jumlah.

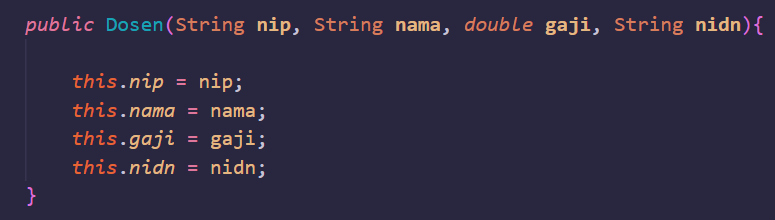
* + 1. Konsep apa dalam OOP yang membolehkan suatu class memiliki constructor atau method dengan nama yang sama dan signature yang berbeda pada satu class?

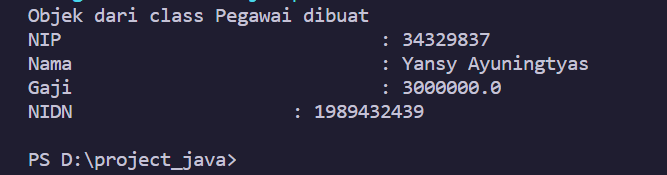
Jawab : Overloading

# PERCOBAAN 7 (super - constructor)

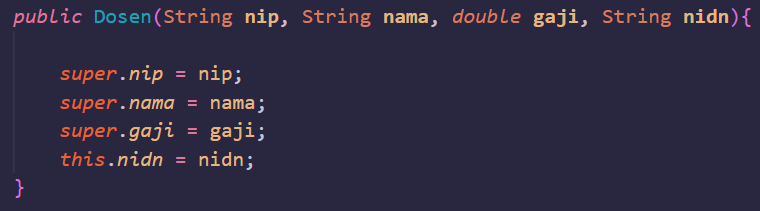
## TAHAPAN PERCOBAAN

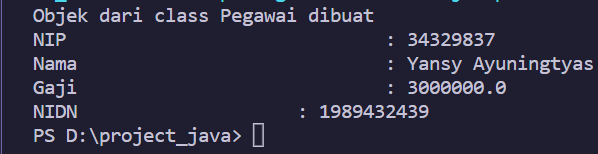
* + 1. Modifikasi constructor pada class Dosen sebagai berikut. Run program kemudian amati hasilnya.



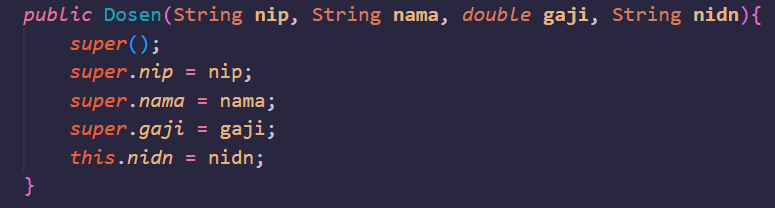


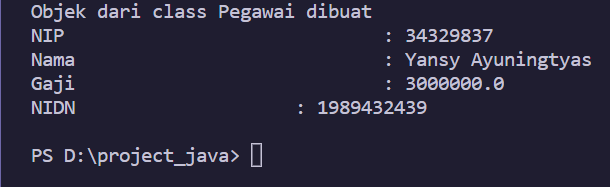
* + 1. Modifikasi constructor pada class Dosen sebagai berikut. Run program kemudian amati hasilnya.



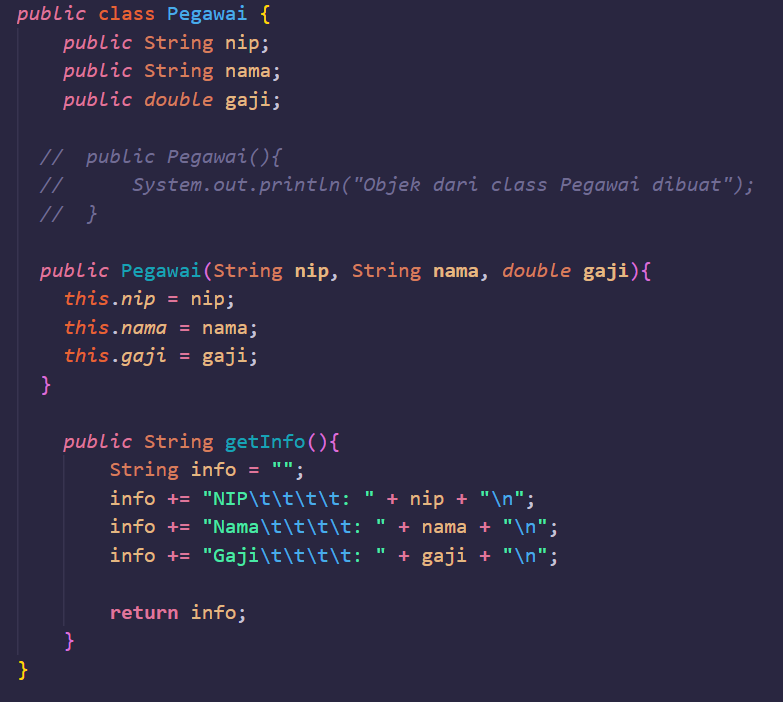


* + 1. Modifikasi constructor pada class Dosen sebagai berikut. Run program kemudian amati hasilnya.

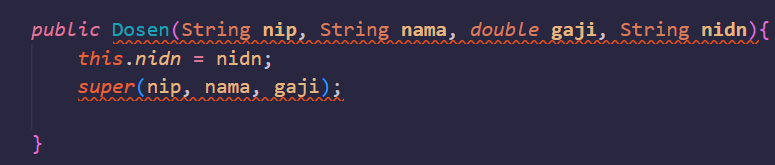


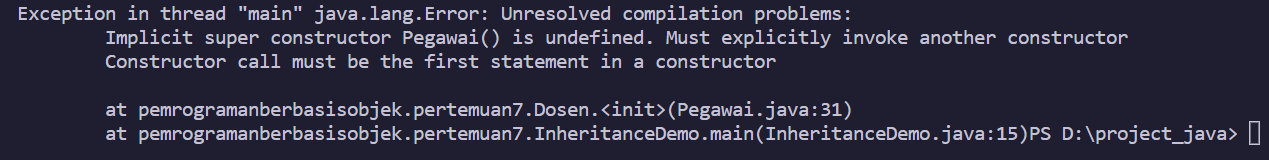


* + 1. Hapus/comment constructor tanpa parameter dari class Pegawai. Tambahkan constructor baru untuk class Pegawai sebagai berikut. Run program kemudian amati hasilnya.

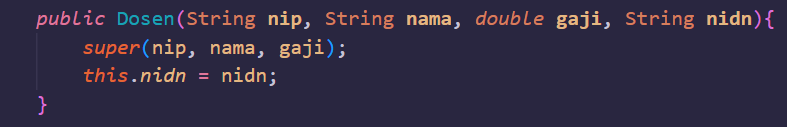


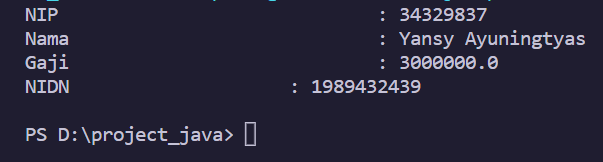
* + 1. Modifikasi constructor pada class Dosen sebagai berikut. Run program kemudian amati hasilnya.





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





## PERTANYAAN

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

Jawab : Tidak ada, semua menampilkan hasil yang sama

* Pada Langkah 1 menggunakan keyword **this, this**  hanya akan langsung merujuk pada subclass sedang subclass otomatis mewarisi atribut dan method dari parent classnya
* Pada Langkah 2 menggunakan keyword **super, super** akan langsung merujuk pada parent class kemudian parent class meng-inherit atribut dan method terhadap subclassnya

Dengan demikian akan menampilkan hasil yang sama dari Langkah 1 dan 2.

* + 1. Apakah terdapat perbedaan hasil pada langkah 2 dan 3? Jelaskan!

Jawab : Tidak, semua menampilkan hasil yang sama karena semua menggunakan keyword **super** perbedaanya pada Langkah 3 ada pemanggilan method constructor yang merujuk pada constructor parent class, yaitu **super();**

* + 1. Mengapa terjadi error pada langkah 4?

Jawab : karena dalam pemanggilan constructor lain harus secara eksplisit sedangkan constructor pada class Pegawai (super constructor) belum ditentukan atau masih secara implisit.

* + 1. Apa perbedaan super() yang dipanggil pada langkah 3 dan 6?

Jawab : pada Langkah 3 keyword **super,** yaitu sintaks **super()**; tidak ada argumennya, masing-masing atribut dideklarasikan menggunakan keyword super untuk dapat merujuk pada parent classnya, kecuali atribut nidn. Sedangkan pada Langkah 6 **super();** diisikan argument untuk langsung mengakses atribut terhadap parent classnya menjadi **super(nip, nama, gaji);** .

* + 1. Mengapa terjadi error pada langkah 5?

Jawab : urutan pemanggilan constructor harus dilakukan pada statement pertama di sebuah constructor

# 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. Lakukan method overriding dengan membuat method dengan nama dan signature yang sama pada parent class dan child class.
  5. Lakukan instansiasi objek child class pada main class kemudian print info nya.