# JOBSHEET W07

**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 {

...

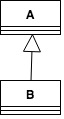
}

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

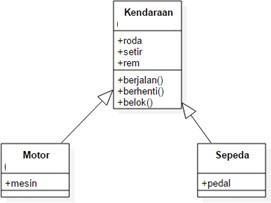
Multilevel inheritance adalah inheritance dengan subclass yang menjadi superclass bagi classyang lain.

Contoh:

Gambar 2. Contoh Multilevel Inheritance

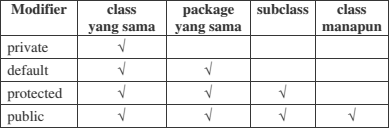
Pada Gambar 2 di atas dapat dilihat bahwa class B merupakan subclass dari class A, namun dia jugamerupakan superclass dari class C. Inheritance dengan 2 level ini disebut multilevel inheritance.

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

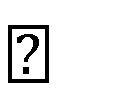
### super(parameter1, parameter2,dst)

Memanggil constructor parent class dengan parameter, hanya dapat dilakukan pada baris pertama dalamconstructor child

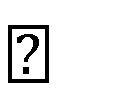
Saat instansiasi objek dari subclass dilakukan, objek pada superclass juga akan terbentuk. Dengan kata lain, ketika constructor subclass dijalankan, pada “baris pertama” (atau sebelum baris- baris lainnya dalam constructor subclass dieksekusi) constructor superclass akan dijalankan terlebih dahulu.

**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, tetapimemiliki 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 dalamhierarki pewarisan.
    - Karakteristik overloading: nama method sama, method signature berbeda, return type bolehsama atau berbeda.
    - JVM menentukan method mana yang akan dipanggil pada compile-time compile- timepolymorphism
    - Disebut juga static binding atau early binding

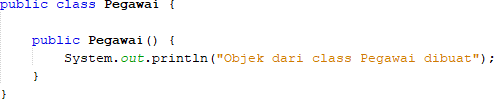
### Overriding

* + - Overriding terjadi ketika child class memiliki method dengan nama dan signature yang samadengan 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

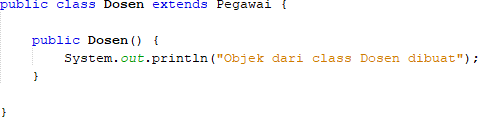
# PERCOBAAN 1 (extends)

## TAHAPAN PERCOBAAN

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



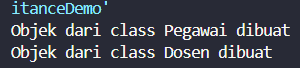
* 1. Buatlah subclass dari class Pegawai dengan nama Dosen, kemudian buat juga constructortanpa parameter dengan baris kode berikut:



* 1. Buatlah main class, misal InheritanceDemo.java, lakukan instansiasi objek baru bernamadosen1 dari class Dosen sebagai berikut:



* 1. Run programnya kemudian amati hasilnya.



## PERTANYAAN

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

Jawab : Parent class nya adalah class Pegawai dan Child classnya adalah Dosen

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

Jawab : kata kunci yang menghubungkan child class dan parent class adalah **Extends**



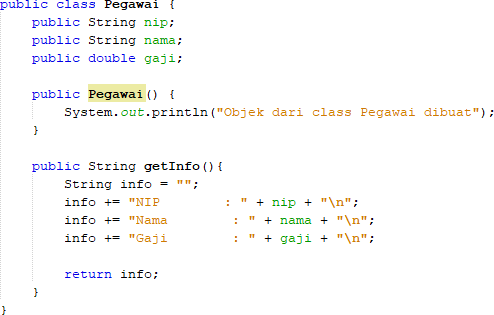
* 1. Berdasarkan hasil yang ditampilkan oleh program, ada berapa constructor yang dieksekusi?Constructor class mana yang lebih dulu dieksekusi?

Jawab : ada 2 constructor yang akan dieksekusi, yang lebih dulu dieksekusi adalah constructor pada superclass setelah itu baru subclassnya karena supaya subclass dapat bekerja dengan benar, atribut dan method yang diwarisi dari superclass harus diinisialisasi terlebih dahulu.

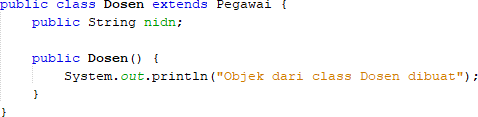
# PERCOBAAN 2 (Pewarisan)

## TAHAPAN PERCOBAAN

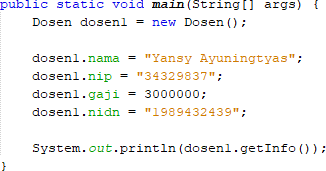
* + - 1. Tambahkan atribut nip, nama, dan gaji serta method getInfo() pada class Pegawai



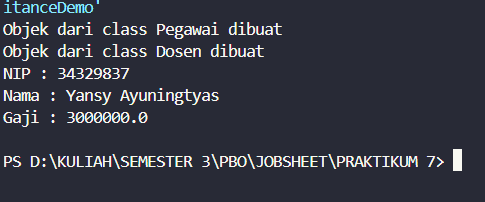
* + - 1. Tambahkan pula atribut NIDN pada class Dosen



* + - 1. Pada class InheritanceDemo.java tuliskan baris kode berikut:



* + - 1. Run program kemudian amati hasilnya



## PERTANYAAN

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

Jawab : Pada percobaan 2 program berhasil dijalankan dan tidak terjadi eror

* + - 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 pada class dosen merupakan subclass dari class Pegawai yang dimana pada class Dosen juga mempunyai atribut dan method yang sama dengan class Pegawai.

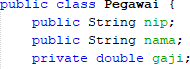
* + - 1. Jika program berhasil dijalankan, mengapa tidak terjadi error pada pemanggilan method getInfo() oleh object dosen1 padahal tidak ada deklarasi method getInfo() pada class Dosen?

Jawab : Karena method getInfo() juga diwariskan dari superclass yaitu class Pegawai.

# PERCOBAAN 3 (Hak akses)

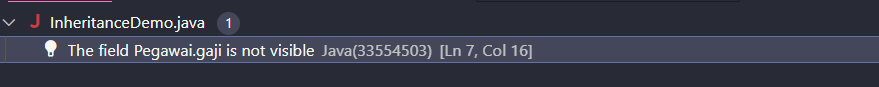
## TAHAPAN PERCOBAAN

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

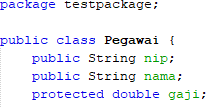


* + 1. Run program kemudian amati hasilnya.

Setelah di run muncul pesan eror seperti berikut.



* + 1. Ubah access level modifier atribut gaji menjadi protected kemudian pindah class Pegawai kepackage baru, misalnya “testpackage”.



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



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



* + 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 akses modifier atribut Private, yang dimana hanya bisa diakses dari class yang sama. Karena gaji memiliki akses private class Dosen yang merupakan subclass dari Pegawai tidak dapat mengakses atribut gaji secara langsung

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

Jawab: karena atribut gaji hanya bisa diakses pada class di package yang sama atau subclass di package yang berbeda

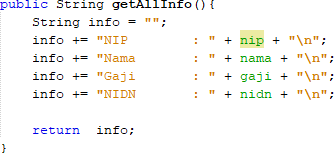
* + 1. Berdasarkan percobaan tersebut, bagaimana menentukan atribut dan method yang akan diwariskan oleh parent class ke child class?

Jawab: menggunakan akses level modifier

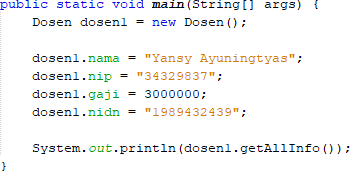
# PERCOBAAN 4 (Super - atribut)

## TAHAPAN PERCOBAAN

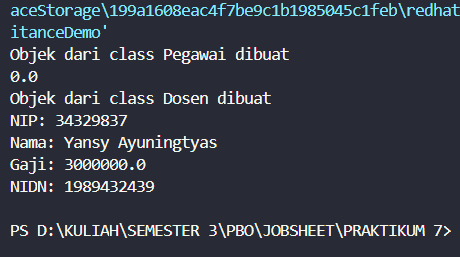
* + 1. Butlah method getAllInfo() pada class Dosen



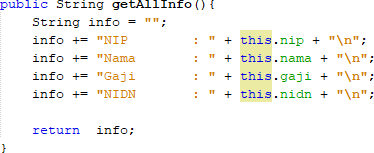
* + 1. Lakukan pemanggilan method getAllInfo() oleh object dosen1 pada class InheritanceDemo.java



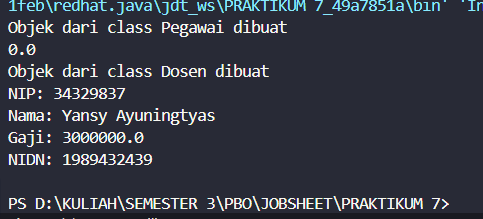
* + 1. Run program kemudian amati hasilnya



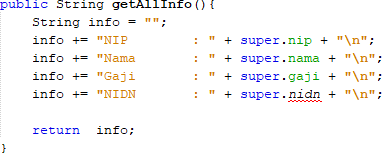
* + 1. Lakukan modifikasi method getAllInfo() pada class Dosen



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

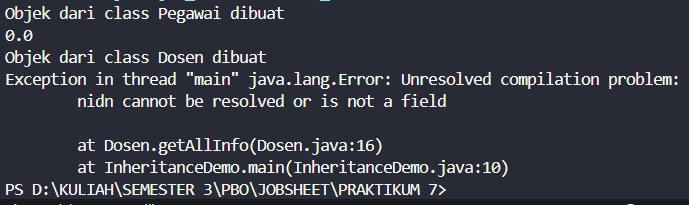


* + 1. Lakukan modifikasi method getAllInfo() pada class Dosen kembali

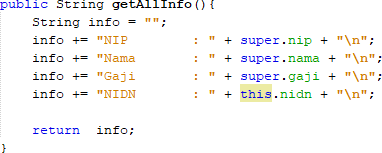


* + 1. Run program kemudian bandingkan hasilnya dengan progam pada no 1 dan no 4.

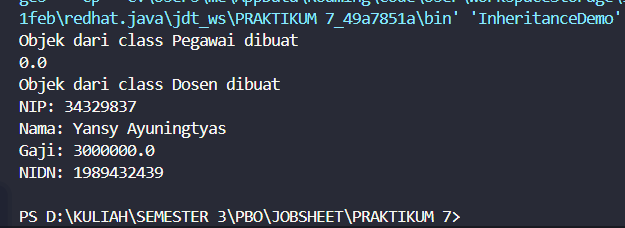
Terjadi eror ketika program di run



* + 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 : Tidak ada perbedaan, **this** merujuk pada instance dari class Dosen (yang mewarisi atribut dari Pegawai), dan **super** merujuk secara eksplisit pada class induk (Pegawai), jadi perbedaannya hanya terlihat pada cara pengaksesan atributnya, namun hasilnya sama

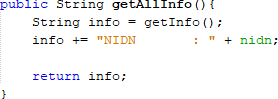
* + 1. Mengapa error terjadi pada program no 6?

Jawab : Pada program no 6 terjadi eror karena atribut nidn bukan berasal dari Parent class, maka dari itu atribut nidn hanya di deklarasikan di class Dosen.

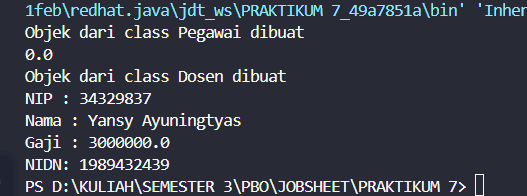
# PERCOBAAN 5 (super & overriding)

## TAHAPAN PERCOBAAN

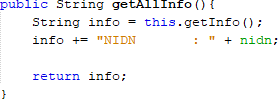
* + 1. Lakukan modifikasi kembali pada method getAllInfo(). Run program kemudian amatihasilnya



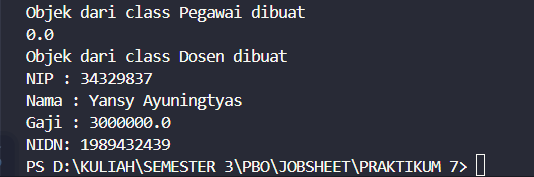
Berikut adalah hasilnya



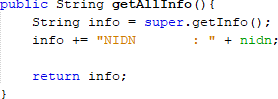
* + 1. Lakukan modifikasi kembali pada method getAllInfo(). Run program kemudian amatihasilnya



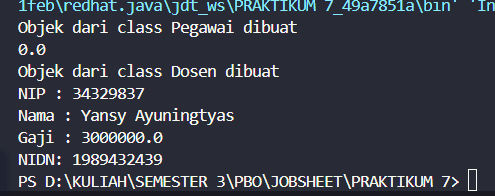
Berikut ini adalah hasilnya



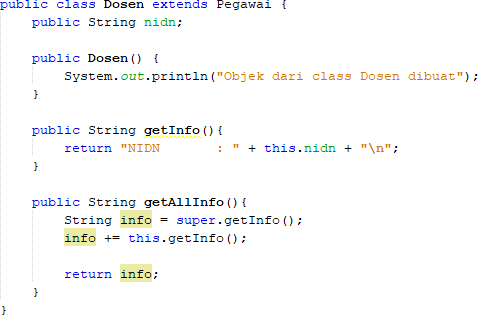
* + 1. Lakukan modifikasi kembali pada method getAllInfo(). Run program kemudian amatihasilnya



Berikut ini adalah 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?

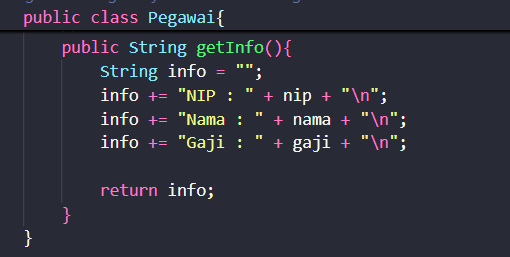
Jawab : tidak ada perbedaan dari langkah 1,2, dan 3

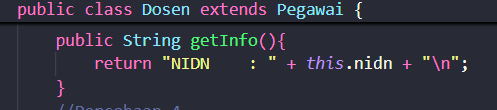
* + 1. Apakah ada perbedaan method super.getInfo() dan this.getInfo() yang dipanggil dalam method getAllInfo() pada langkah 4? Jelaskan!

Jawab : Ketika super.getInfo() dipanggil dalam method getAllInfo, maka akan secara langsung memanggil implementasi getInfo() yang ada pada Superclass

* + 1. Pada method manakah terjadi overriding? Jelaskan!

Jawab : Overriding terjadi ketika subclass memiliki method dengan nama dan signature yang sama dengan superclassnya. Pada class Dosen terdapat method getInfo() dengan signature yang sama dengan class Pegawai





* + 1. Tambahkan keyword final pada method getInfo() di class Pegawai. Apakah program dapat dicompile? Mengapa?

Jawab : karena program tidak dapat dicompile karena final merupakan modifier yang digunakan untuk membuat class, atribut atau method tidak dapat diubah

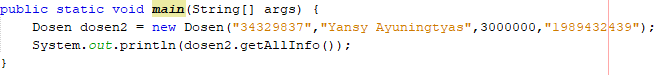
# PERCOBAAN 6 (overloading)

## 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 yang ditampilkan berupa nilai null

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

Jawab : tidak memiliki signature yang sama, karena jika kedua constructor tersebut memiliki signature yang sama maka akan terjadi error pada saat kompilasi.

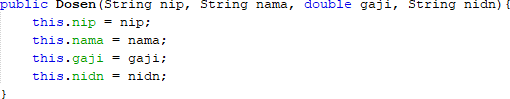
* + 1. Konsep apa dalam OOP yang membolehkan suatu class memiliki constructor atau method dengan nama yang sama dan signature yang berbeda pada satu class?

Jawab : pada konsep **OVERLOADING**

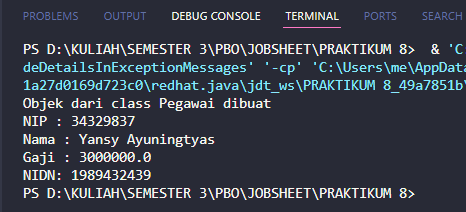
# PERCOBAAN 7 (super - constructor)

## TAHAPAN PERCOBAAN

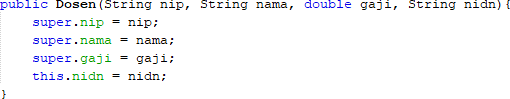
* + 1. Modifikasi constructor pada class Dosen sebagai berikut. Run program kemudian amatihasilnya.

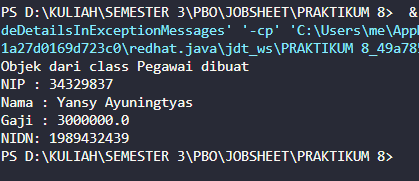


Hasilnya :



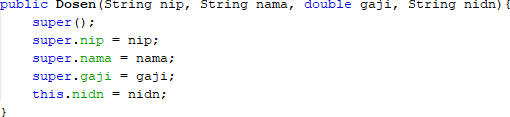
* + 1. Modifikasi constructor pada class Dosen sebagai berikut. Run program kemudian amatihasilnya.



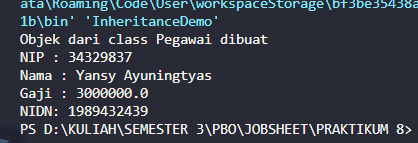


* + 1. Modifikasi constructor pada class Dosen sebagai berikut. Run program kemudian

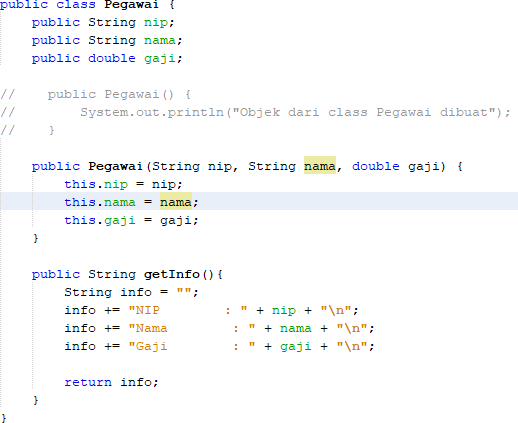
amatihasilnya.

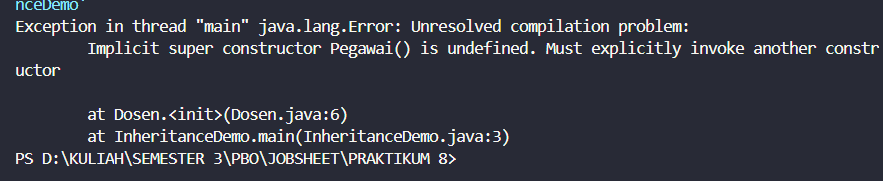


Hasilnya :

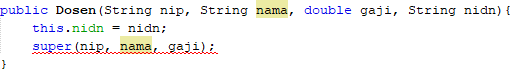


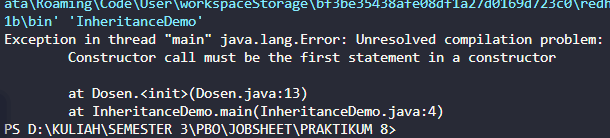
* + 1. Hapus/comment constructor tanpa parameter dari class Pegawai. Tambahkan constructor baru untuk class Pegawai sebagai berikut. Run program kemudian amati hasilnya.



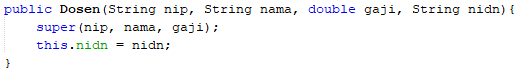


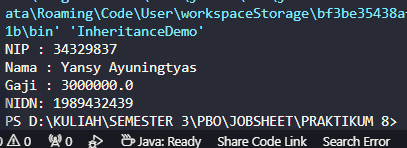
* + 1. Modifikasi constructor pada class Dosen sebagai berikut. Run program kemudian amatihasilnya.





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





## PERTANYAAN

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

Jawab : Pada langkah 1 menggunakan **this** yang dimana merujuk pada atribut pada current class/object. Jika tidak maka baru akan merujuk ke atribut parent class. Sedangkan langkah 2 menggunakan kata kunci **super** yang merujuk langsung pada parent class/object

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

Jawab : Perbedaannya adalah

* **Langkah 2**: Constructor menggunakan **super.nip = nip;**, **super.nama = nama;**, dan **super.gaji = gaji;** untuk mengisi atribut dari superclass **Pegawai** secara langsung.
* **Langkah 3**: Constructor diubah menggunakan **super();** dan tetap menggunakan **super.nip = nip;**, **super.nama = nama;**, dan **super.gaji = gaji;**, namun memanggil constructor default dari superclass dengan super().
  + 1. Mengapa terjadi error pada langkah 4?

Jawab : Error terjadi karena **constructor default (tanpa parameter)** pada superclass **Pegawai** tidak didefinisikan.

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

Jawab : pada langkah 3 memanggil constructor default dari superclass, sedangkan pada langkah 6 yang dipanggil adalah constructor dari superclass yang memiliki parameter

* + 1. Mengapa terjadi error pada langkah 5?

Jawab : karena kata kunci super dipanggil setelah this, yang dimana seharusnya kata kunci super dipanggil terlebih dahulu daripada this.

# 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 2 objek child class pada main class kemudian print info nya.

Parent Class



Sub class





Class Main



Hasil runnya

