**Maritza Ulfa Huriyah**

**2241760119**

**SIB 2C**

# 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 {

...

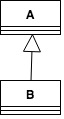
}

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

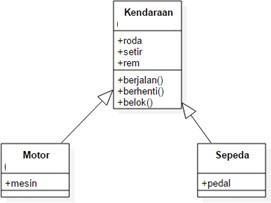
Multilevel inheritance adalah inheritance dengan subclass yang menjadi superclass bagi classyang lain.

Contoh:

Gambar 2. Contoh Multilevel Inheritance

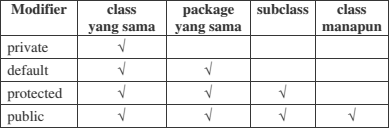
Pada Gambar 2 di atas dapat dilihat bahwa class B merupakan subclass dari class A, namun dia jugamerupakan superclass dari class C. Inheritance dengan 2 level ini disebut multilevel inheritance.

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

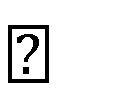
### super(parameter1, parameter2,dst)

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

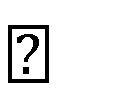
Saat instansiasi objek dari subclass dilakukan, objek pada superclass juga akan terbentuk. Dengan kata lain, ketika constructor subclass dijalankan, pada “baris pertama” (atau sebelum baris- baris lainnya dalam constructor subclass dieksekusi) constructor superclass akan dijalankan terlebih dahulu.

**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

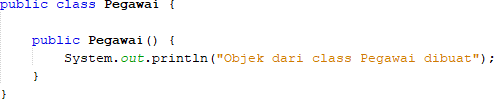
### Overriding

* + - 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

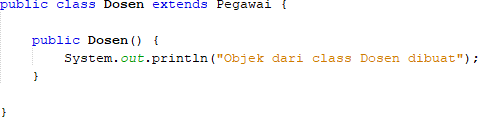
# PERCOBAAN 1 (extends)

## TAHAPAN PERCOBAAN

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



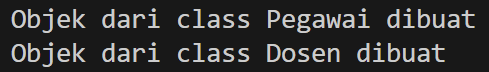
* 1. Buatlah subclass dari class Pegawai dengan nama Dosen, kemudian buat juga constructortanpa parameter dengan baris kode berikut:



* 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!
* Parent class: Pegawai
* Child class: Dosen
  1. Kata kunci apa yang membuat child class dan parent class tersebut memiliki relasi?
* Kata kunci yang membuat child class dan parent class memiliki relasi yaitu “extends”, yang ada pada kode:

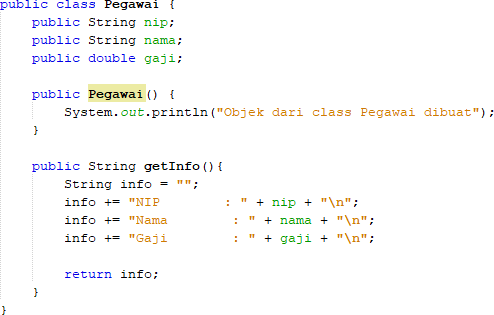
public class Dosen **extends** Pegawai {

* 1. Berdasarkan hasil yang ditampilkan oleh program, ada berapa constructor yang dieksekusi?Constructor class mana yang lebih dulu dieksekusi?
* Konstruktor class Pegawai akan lebih dulu di eksekusi oleh program, lalu setelah itu baru mengeksekusi kontruktor di class Dosen

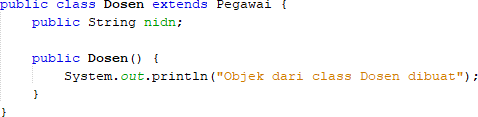
# PERCOBAAN 2 (Pewarisan)

## TAHAPAN PERCOBAAN

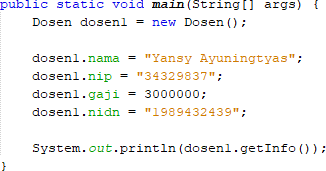
* + - 1. Tambahkan atribut nip, nama, dan gaji serta method getInfo() pada class Pegawai



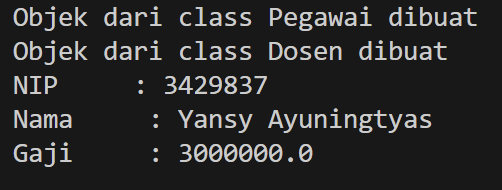
* + - 1. Tambahkan pula atribut NIDN pada class Dosen



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



* + - 1. Run program kemudian amati hasilnya



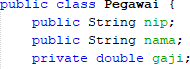
## PERTANYAAN

* + - 1. Pada percobaan 2 diatas, apakah program dapat berhasil dijalankan ataukah terjadi error?
* Program diatas berhasil dijalankan
  + - 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?
* Karena instansiasi doseni1 berdasarkan class Dosen. Sedangkan class Dosen merupakan child class dari class Pegawai. Jadi attribut pada class Pegawai diturunkan kepada class Dosen. Begitu juga dengan method nya, salah satunya adalah getInfo() yang medeklarasikan 3 attribut tersebut. Itu mengapa ketika dipanggil tidak error walau belum di deklarasikan pada class Dosen, karen sudah dideklarasikan pada 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?
* Karena pada class Dosen tidak ada, maka pemanggilan method getInfo() dari parent classnya, yaitu class Pegawai.

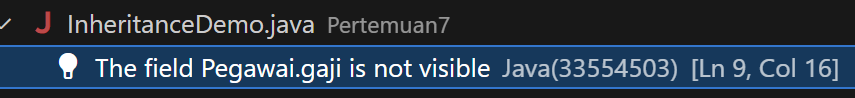
# PERCOBAAN 3 (Hak akses)

## TAHAPAN PERCOBAAN

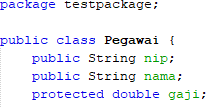
* + 1. Modifikasi access level modifier pada atribut gaji menjadi private pada class Pegawai.java



* + 1. Run program kemudian amati hasilnya.



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



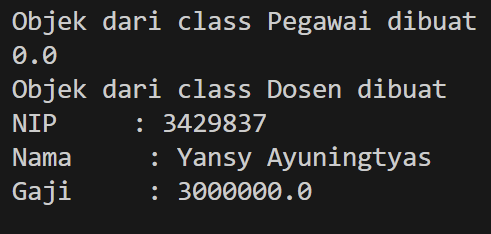
* + 1. Import class Pegawai dari testpackage pada class Dosen.



* + 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.



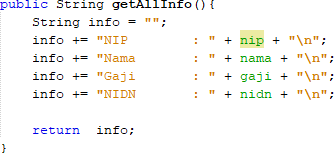
## PERTANYAAN

* + 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?
* Karena attribut gaji access modifier nya diubah menjadi private, yang membuat atribut tidak bisa diakses diluar kelas Pegawai.
  + 1. Pada langkah 5, setelah class Pegawai berpindah ke package yang berbeda, class Dosen masih dapat mengakses atribut gaji. Mengapa?
* Karena access modifier protected memungkinkan untuk pemanggilan diluar packagenya.
  + 1. Berdasarkan percobaan tersebut, bagaimana menentukan atribut dan method yang akan diwariskan oleh parent class ke child class?
* Atribut dan method yang diwariskan dari parent class ke child class ada alah yang tida bersifat private.

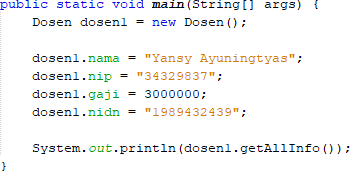
# PERCOBAAN 4 (Super - atribut)

## TAHAPAN PERCOBAAN

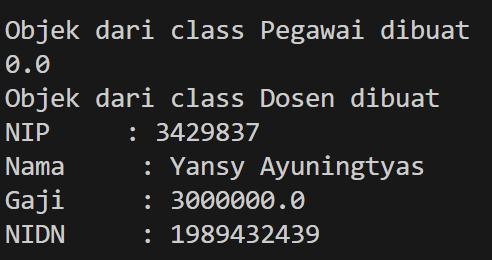
* + 1. Butlah method getAllInfo() pada class Dosen



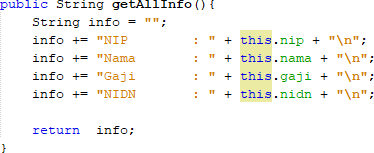
* + 1. Lakukan pemanggilan method getAllInfo() oleh object dosen1 pada class InheritanceDemo.java



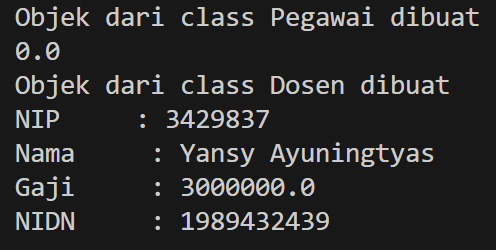
* + 1. Run program kemudian amati hasilnya



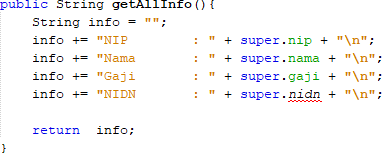
* + 1. Lakukan modifikasi method getAllInfo() pada class Dosen



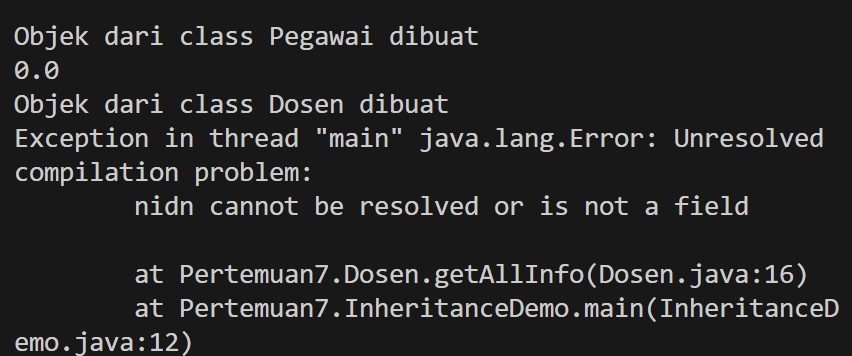
* + 1. Run program kemudian bandingkan hasilnya dengan langkah no 2.



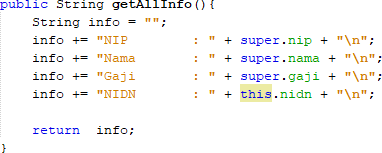
* + 1. Lakukan modifikasi method getAllInfo() pada class Dosen kembali



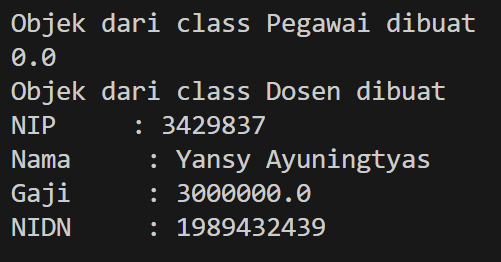
* + 1. Run program kemudian bandingkan hasilnya dengan progam pada no 1 dan no 4.



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



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



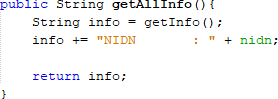
## PERTANYAAN

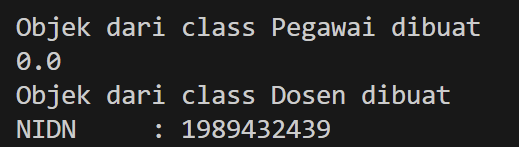
* + 1. Apakah terdapat perbedaan hasil nama, nip, dan gaji yang ditampilkan pada program 1,4,dan 8? Mengapa?
* Pada program nomor 1, kode bisa dijalankan. Karena merujuk yang terdekat. Pada kasus nomor 1 merujuk pada atribut class nya
* Pada program nomor 4, kode “this” merujuk pada current class atau pada class Dosen.
* Pada program no 8, kode “super” merujuk pada atribut di parent class atau class Pegawai
  + 1. Mengapa error terjadi pada program no 6?
* Karena ketik amemakai kode “super” maka akan langsung merujuk ke parent class. Namun atribut nidn berada pada current class/class Dosen tersebut.

# PERCOBAAN 5 (super & overriding)

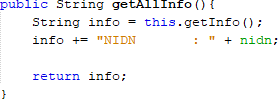
## TAHAPAN PERCOBAAN

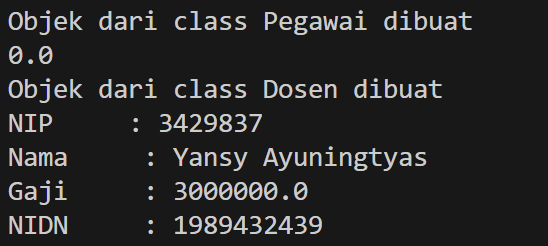
* + 1. Lakukan modifikasi kembali pada method getAllInfo(). Run program kemudian amatihasilnya



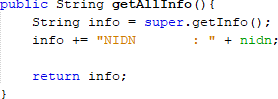


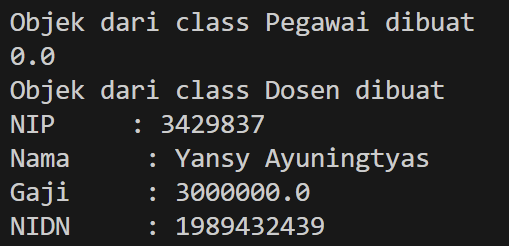
* + 1. Lakukan modifikasi kembali pada method getAllInfo(). Run program kemudian amatihasilnya



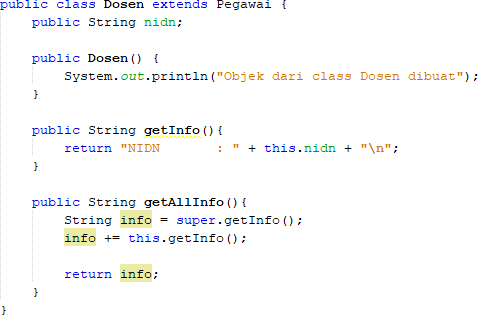


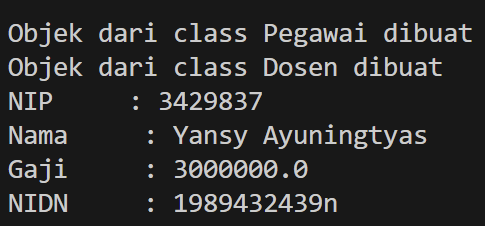
* + 1. Lakukan modifikasi kembali pada method getAllInfo(). Run program kemudian amatihasilnya





* + 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?
* Pada program 1, atribut yang lain tidak keluar pada output karena tidak memanggil atribut dari parent class. Sedangkan pada program 2 dan 3, merujuk pada atribut di prent class.
  + 1. Apakah ada perbedaan method super.getInfo() dan this.getInfo() yang dipanggil dalam method getAllInfo() pada langkah 4? Jelaskan!
* Pada method super.getInfo(), yang dipanggil adalah method pada parent class atau class Pegawai. Sedangkan pada methos this.getInfo() yang dipanggil adalah getInfo() pada class tersebut(class Dosen).
  + 1. Pada method manakah terjadi overriding? Jelaskan!
* Overriding pada kode tersebut terjadi pada method getInfo(), karena method getInfo() pada class Dosen menggantikan method getInfo() pada class Pegawai.
  + 1. Tambahkan keyword final pada method getAllInfo() di class Pegawai. Apakah program dapat dicompile? Mengapa?
* Tidak dapat di compile. Dikarenakan setelah diberi final, method tidak dapat diedit dan hanya dapat dipanggil saja. Lalu pada class Dosen terdapat override namun tidak dapat diakses karena adanya final di getAlInfo() di class Pegawai.

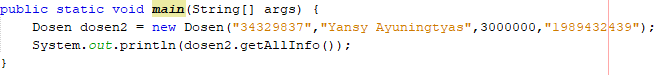
# PERCOBAAN 6 (overloading)

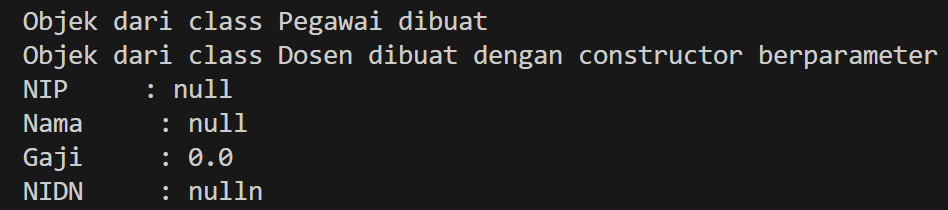
## TAHAPAN PERCOBAAN

* + 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.





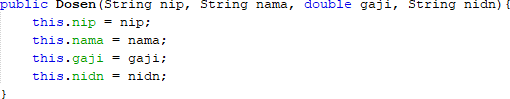
## PERTANYAAN

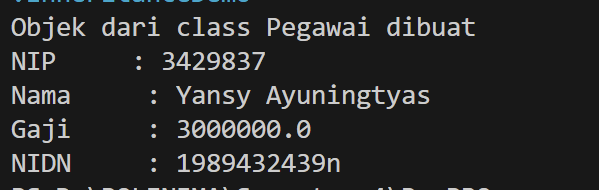
* + 1. Bagaimana hasil nilai nip, nama, gaji, dan nidn yang ditampilkan pada langkah 2?Mengapa demikian?
* Output yang dikeluarkan yaitu null, karena pada kontruktor di class Dosen belum di inisialisasi ke variabel. Jadi belum bisa menampung nilai datanya.
  + 1. Jelaskan apakah constructor tanpa parameter dan constructor class Dosen yang dibuat padalangkah 1 memiliki signature yang sama?
* Tidak, karena susunan dan jumlahnya berbeda.
  + 1. Konsep apa dalam OOP yang membolehkan suatu class memiliki constructor atau methoddengan nama yang sama dan signature yang berbeda pada satu class?
* Konsep dalam OOP yang memperbolehkan suatu class memiliki contructor yang berbeda pada satu class adalah konsep overleading.

# PERCOBAAN 7 (super - constructor)

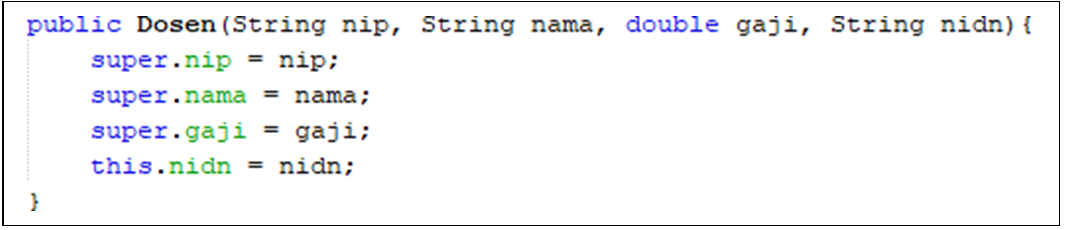
## TAHAPAN PERCOBAAN

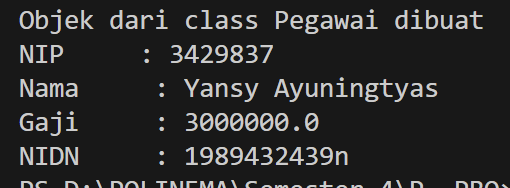
* + 1. Modifikasi constructor pada class Dosen sebagai berikut. Run program kemudian amatihasilnya.



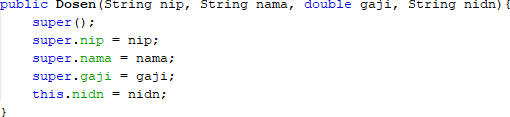


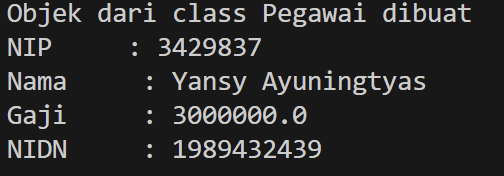
* + 1. Modifikasi constructor pada class Dosen sebagai berikut. Run program kemudian amatihasilnya.



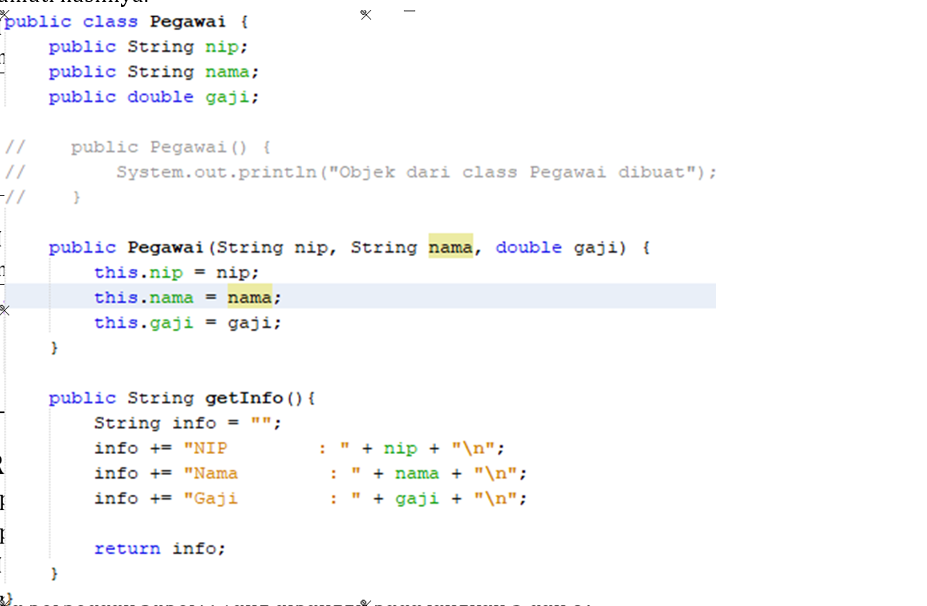


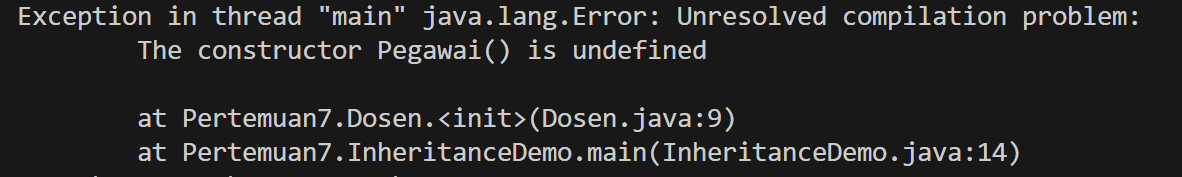
* + 1. Modifikasi constructor pada class Dosen sebagai berikut. Run program kemudian amatihasilnya.



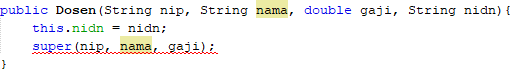


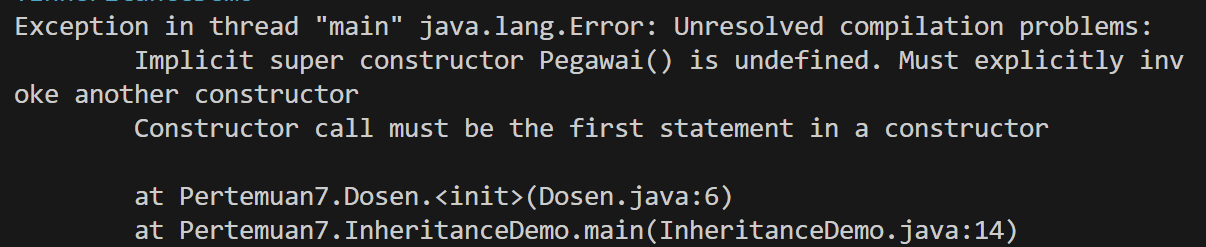
* + 1. Hapus/comment constructor tanpa parameter dari class Pegawai. Tambahkan constructor baru untuk class Pegawai sebagai berikut. Run program kemudian amati hasilnya.



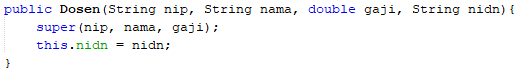


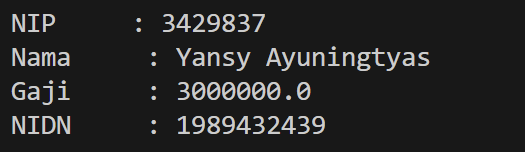
* + 1. Modifikasi constructor pada class Dosen sebagai berikut. Run program kemudian amatihasilnya.





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



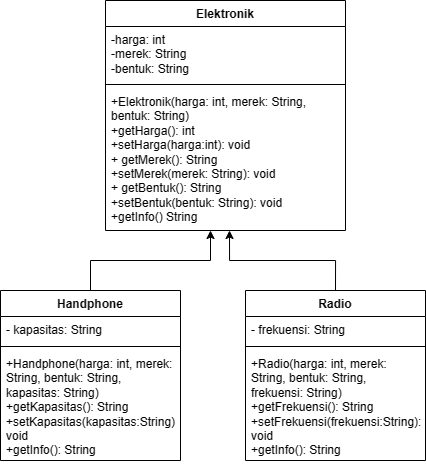


## PERTANYAAN

* + 1. Apakah terdapat perbedaan hasil pada langkah 1 dan 2? Jelaskan!
* Tidak, karena pada langkah 1 maupun 2 semua merujuk pada parent class.
  + 1. Apakah terdapat perbedaan hasil pada langkah 2 dan 3? Jelaskan!
* Tidak ada, karena memakai “super” atau tidak, akan tetap memanggil atribut pada parent class.
  + 1. Mengapa terjadi error pada langkah 4?
* Karena pada super() harusnya menambahkan atribut yang sudah diinisiasi yang ada pada konstruktor class Pegawai.
  + 1. Apa perbedaan super() yang dipanggil pada langkah 3 dan 6?
* Untuk output sama, namun untuk penulisan berbeda. Jika di langkah 6 hanya ada super(), di langkah 6 ada method super() dengan parameter.
  + 1. Mengapa terjadi error pada langkah 5?
* Karena seharusnya pemanggilan konstruktor di dalam konstruktor harus diawal.

# TUGAS

* 1. Tentukan sebuah class yang merupakan turunan dari class yang lain.
* Class Elektronik, class Handphone, dan class Radio
  1. Buat 3 atribut pada parent class kemudian tambahkan minimal 1 atribut pada child class.



* 1. 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.
* Kode class Elektronik

    public Elektronik(){

    }

    public Elektronik(String merek){

    }

    public Elektronik(String merek, int harga, String tipe){

        this.merek=merek;

        this.harga=harga;

        this.tipe=tipe;

    }

* Kode class Handphone

 public Handphone(){

    }

    public Handphone(String merek){

    }

    public Handphone(String merek, int harga, String tipe, String kapasitas){

        super(merek, harga, tipe);

        this.kapasitas=kapasitas;

    }

* Kode class Radio

  public Radio(){

    }

    public Radio(String merek){

    }

    public Radio(String merek, int harga, String tipe, String frekuensi){

        super(merek, harga, tipe);

        this.frekuensi=frekuensi;

    }

* 1. Lakukan method overriding dengan membuat method dengan nama dan signature yang sama pada parent class dan child class.
* Kode class Elektronik:

public Elektronik(String merek, int harga, String tipe){

        this.merek=merek;

        this.harga=harga;

        this.tipe=tipe;

    }

* Kode class Handphone:

public Handphone(String merek, int harga, String tipe, String kapasitas){

        super(merek, harga, tipe);

        this.kapasitas=kapasitas;

    }

* Kode class Radio:

public Radio(String merek, int harga, String tipe, String frekuensi){

        super(merek, harga, tipe);

        this.frekuensi=frekuensi;

    }

* 1. Lakukan instansiasi objek child class pada main class kemudian print info nya.
* Output:

