# JOBSHEET W06

**INHERITANCE**

# 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 atributdan method baik yang dimiliki sendiri atau turunan dari superclass nya.

# 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.

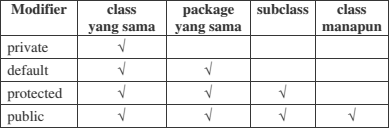
public class B extends A {

...

}

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.

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** digunakan untuk merujuk pada current object/class. Sementara kata kunci **super** digunakan untuk merujuk pada parent object/class. Format penulisannya adalah sebagai berikut:

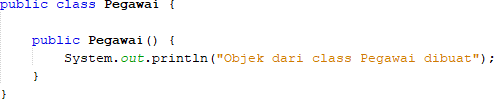
### super.<namaAtribut>

Mengakses atribut parent

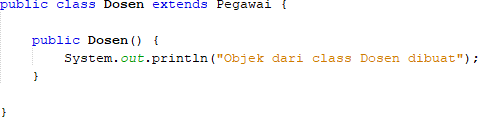
### super.<namaMethod>()

Memanggil method parent

1. **PERCOBAAN 1 (extends)**
   1. **TAHAPAN PERCOBAAN**
      1. Buatlah sebuah parent class dengan nama Pegawai. Lalu buat constructor tanpa parameter dengan baris kode sebagai berikut:



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



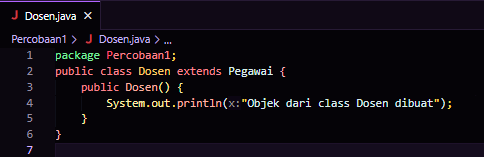
* + 1. Buatlah main class, misal InheritanceDemo.java, lakukan instansiasi objek baru bernama dosen1 dari class Dosen sebagai berikut:



* + 1. Run programnya kemudian amati hasilnya.
  1. **PERTANYAAN**
     1. Pada percobaan 1 diatas, tentukan child class dan parent class!

**The Lecturer and Employee classes are two separate classes that do not show any direct inheritance relationship.**

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



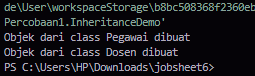
**extends keyword is required. If Lecturer is a child class of Employee**

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

**because adding inheritance using the extends keyword then there will be two constructors executed:**

**The constructor of the Employee class will be executed first (because this class is the parent class).**

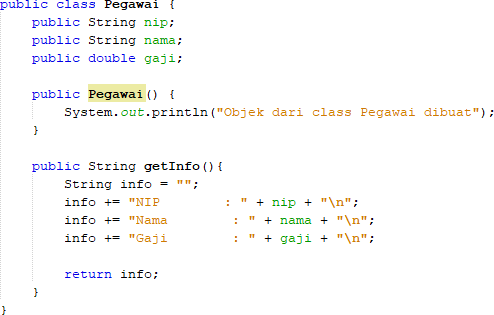
**After that, the constructor of the Dosen class (child class) will be executed**

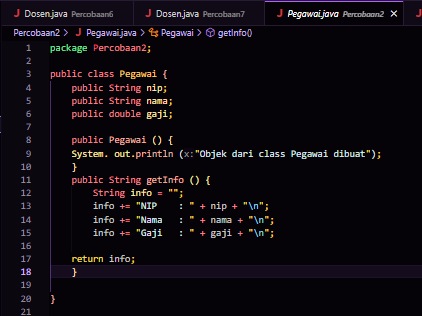
****

# PERCOBAAN 2 (Pewarisan)

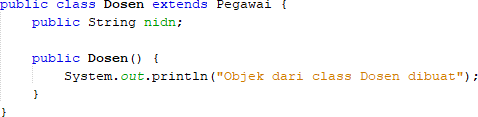
## TAHAPAN PERCOBAAN

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

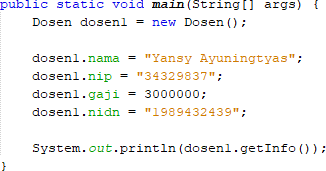


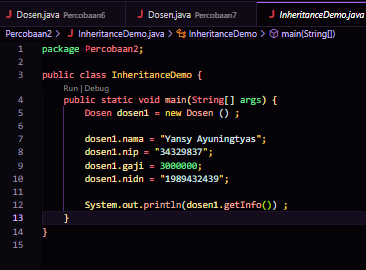


* + - * 1. Tambahkan pula atribut NIDN pada class Dosen



* + - * 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?

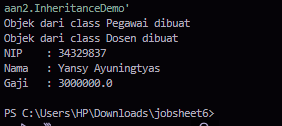
**the program will run successfully and no errors will occur.**

* + - * 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?

**The nip, name, and salary attributes can be accessed by the dosen1 object because the modifier is public, and the nidn attribute is declared directly in the Dosen 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?

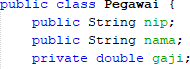
**There is no error when calling the getInfo() method by the lecturer1 object because getInfo() can be called by the lecturer1 object because the Lecturer class inherits this method from the Employee class, and the method is public.**

****

# PERCOBAAN 3 (Hak akses)

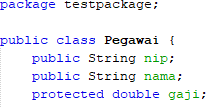
## TAHAPAN PERCOBAAN

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



* + 1. Run program kemudian amati hasilnya.
    2. Ubah access level modifier atribut gaji menjadi protected kemudian pindah class Pegawai

ke package baru, misalnya “testpackage”.



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



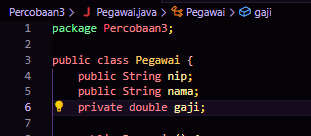
* + 1. Akses atribut gaji pada class Dosen dengan coba mencetak atribut gaji pada constructor Dosen



* + 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?



**because the salary attribute in the Employee class is declared with the private modifier. private can only be accessed in the class where the attribute is declared**

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

**The Dosen class can still access the salary attribute after the Employee class is moved to a different package because the salary is changed to protected. The protected modifier allows access by the class itself, subclasses, and other classes in the same package. Thus, attribute inheritance still works even though it is in a different package.**

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

**The attributes and methods inherited by a child class are determined by its access modifiers:**

**Private is not inherited.**

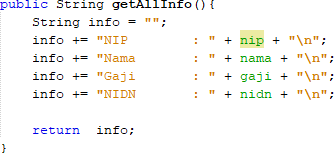
**Protected allows access in subclasses, even in different packages.**

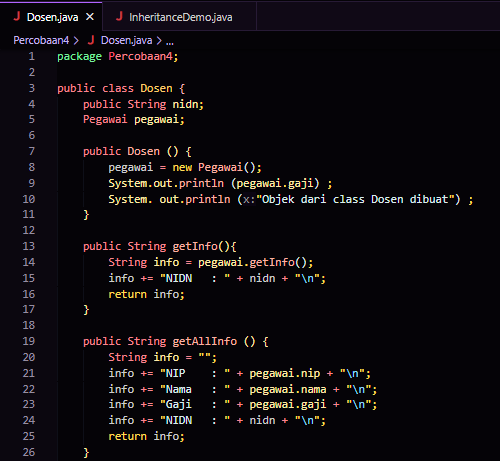
**Public is inherited and can be accessed from anywhere.**

# PERCOBAAN 4 (Super - atribut)

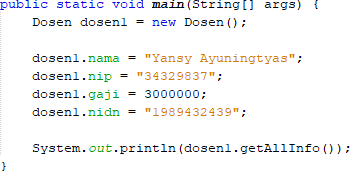
## TAHAPAN PERCOBAAN

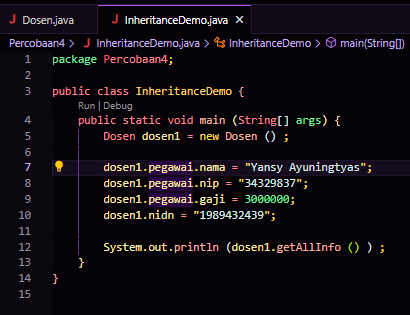
* + 1. Butlah method getAllInfo() pada class Dosen



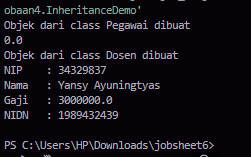


* + 1. Lakukan pemanggilan method getAllInfo() oleh object dosen1 pada class InheritanceDemo.java

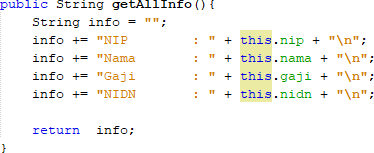




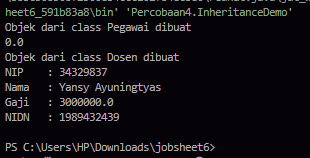
* + 1. Run program kemudian amati hasilnya



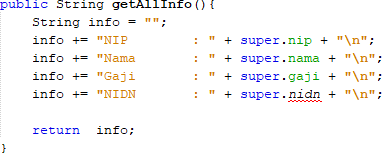
* + 1. Lakukan modifikasi method getAllInfo() pada class Dosen

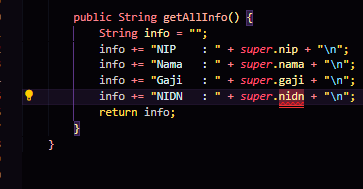


* + 1. Run program kemudian bandingkan hasilnya dengan langkah no 2.

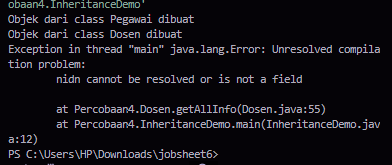


* + 1. Lakukan modifikasi method getAllInfo() pada class Dosen kembali

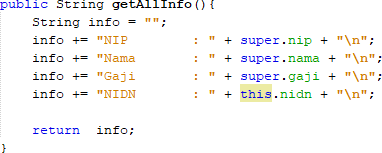


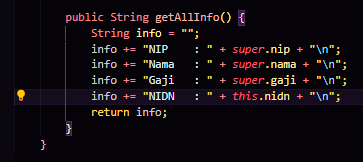


* + 1. Run program kemudian bandingkan hasilnya dengan progam pada no 1 dan no 4.



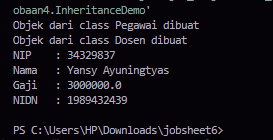
* + 1. Lakukan modifikasi method getAllInfo() pada class Dosen kembali





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

**program 1 and 4 have the same results, all information such as NIP, Name, Salary, and NIDN are displayed correctly. While in program 8, initially there were no values ​​for NIP, Name, Salary, and NIDN displayed.**



## PERTANYAAN

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

**There is a Difference: The results displayed in programs 1 and 4 are the same, while program 8 experiences an error in displaying the initial value.**

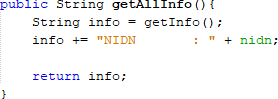
* + 1. Mengapa error terjadi pada program no 6?

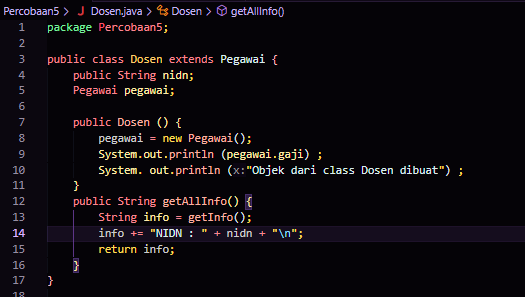
**Incorrect use of super and this**

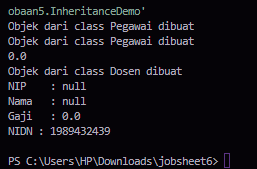
# PERCOBAAN 5 (super & ov e r r id i n g )

## TAHAPAN PERCOBAAN

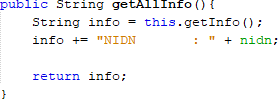
* + 1. Lakukan modifikasi kembali pada method getAllInfo(). Run program kemudian amati hasilnya

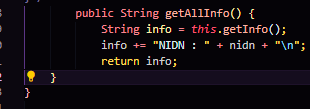


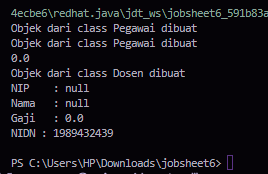




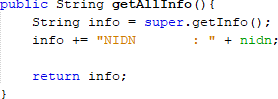
* + 1. Lakukan modifikasi kembali pada method getAllInfo(). Run program kemudian amati hasilnya

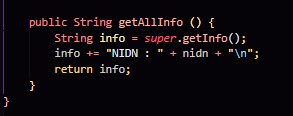


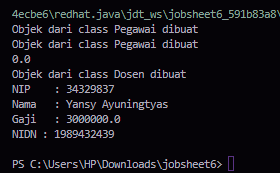




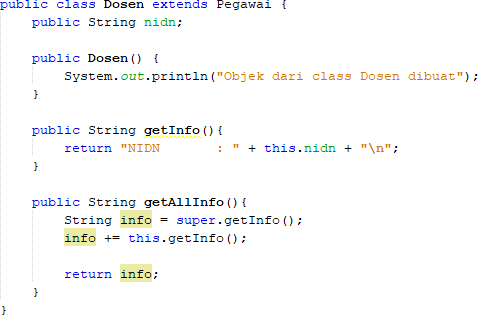
* + 1. Lakukan modifikasi kembali pada method getAllInfo(). Run program kemudian amati hasilnya

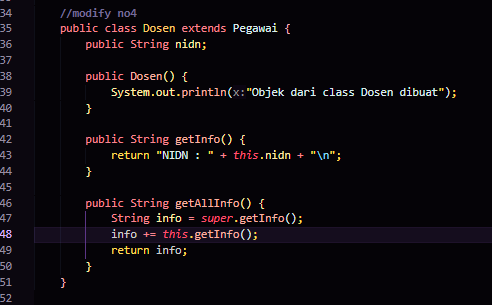


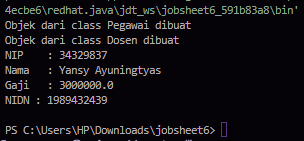




* + 1. Tambahkan method getInfo() pada class Dosen dan modifikasi method getAllInfo() sebagai berikut







## PERTANYAAN

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

**Step 1 and 2: Calling getInfo() from Lecturer without setting the attribute, so the result is the default value.**

**Step 3: Calling super.getInfo() but because the Lecturer attribute value is already set, the correct information is displayed.**

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

**The main difference is where the methods are called from. super.getInfo() accesses a method from the parent class, while this.getInfo() accesses a method from the current class (Dosen), which means it returns more specific information.**

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

**Overriding: Is a process in which a subclass (in this case, Lecturer) provides a new implementation for a method that has been defined in the superclass (in this case, Employee).**

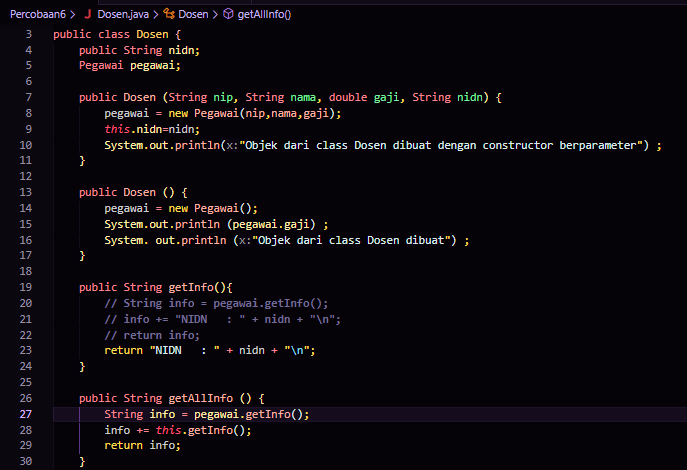
**Overriding occurs in the getInfo() method in the Dosen class.**

# PERCOBAAN 6 (overloading)

## TAHAPAN PERCOBAAN

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

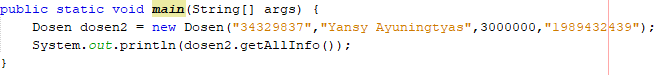


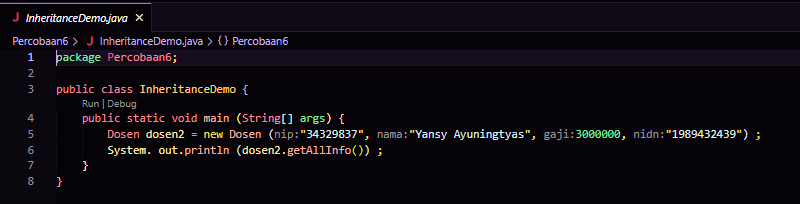


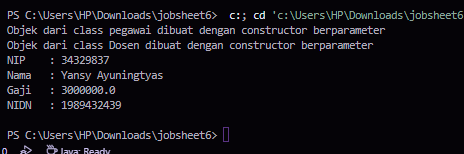
* + 1. Modifikasi class InheritanceDemo untuk menginstansiasi object baru dengan nama dosen2 dengan constructor yang berparameter. Run program kemudian amati hasilnya.

**The parameterized constructor in the Dosen class is used to directly set the values ​​of NIP, Name, Salary, and NIDN when the object is created.**

**Create a dosen2 object with a parameterized constructor and display complete lecturer information using the getAllInfo() method.**







## PERTANYAAN

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

**The NIP, Name, Salary, and NIDN values ​​will be displayed according to the parameters given when creating the dosen2 object. This is because the constructor with parameters directly initializes these values ​​when the object is created.**

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

**The parameterless constructor and the constructor created in step 1 do not have the same signature. The signatures are different because the parameterless constructor does not accept arguments, while the constructor in step 1 accepts four parameters (nip, name, salary, and nidn).**

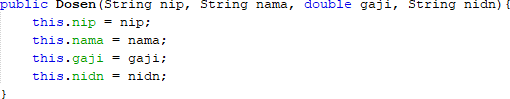
* + 1. Konsep apa dalam OOP yang membolehkan suatu class memiliki constructor atau method dengan nama yang sama dan signature yang berbeda pada satu class?

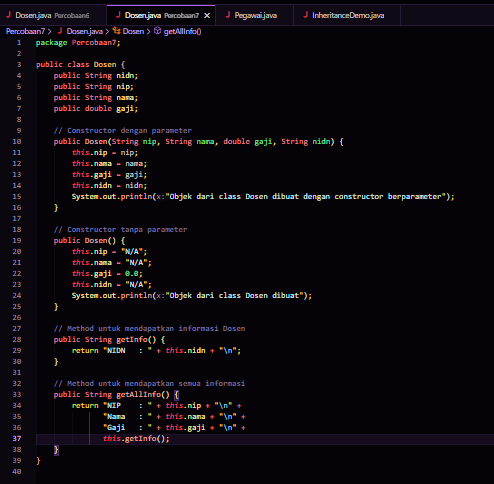
Overloading allows a class to have multiple methods or constructors with the same name, but with different signatures (number or type of parameters).

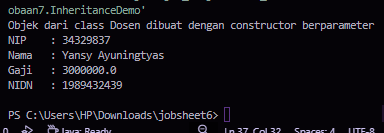
# PERCOBAAN 7 (super - constructor)

## TAHAPAN PERCOBAAN

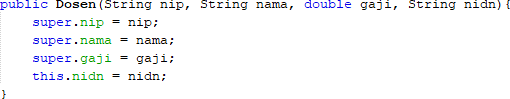
* + 1. Modifikasi constructor pada class Dosen sebagai berikut. Run program kemudian amati hasilnya.

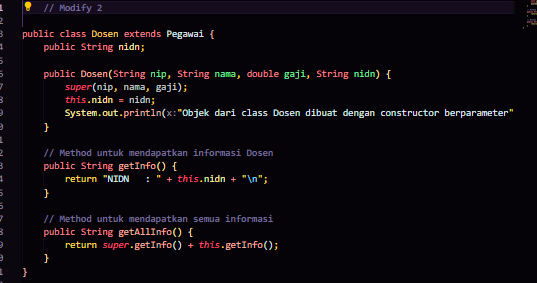


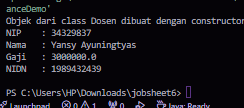




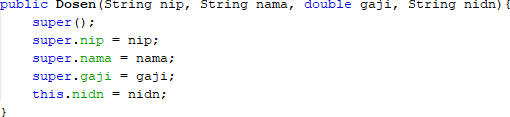
* + 1. Modifikasi constructor pada class Dosen sebagai berikut. Run program kemudian amati hasilnya.

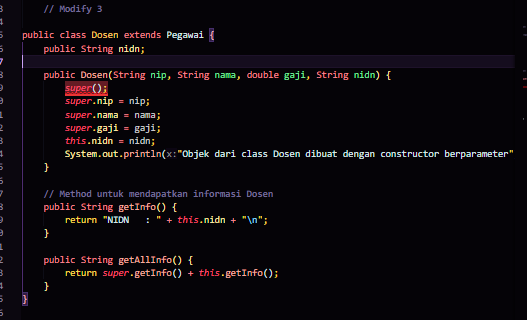


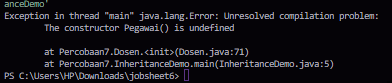




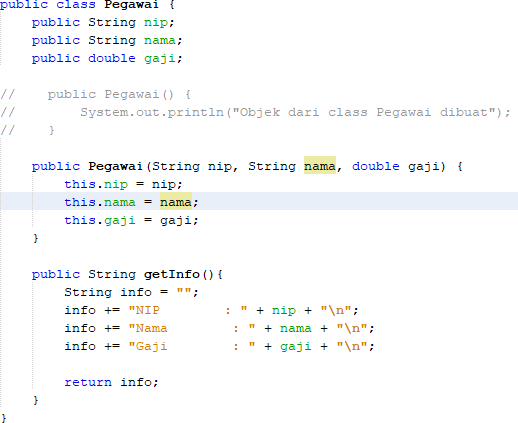
* + 1. Modifikasi constructor pada class Dosen sebagai berikut. Run program kemudian amati hasilnya.

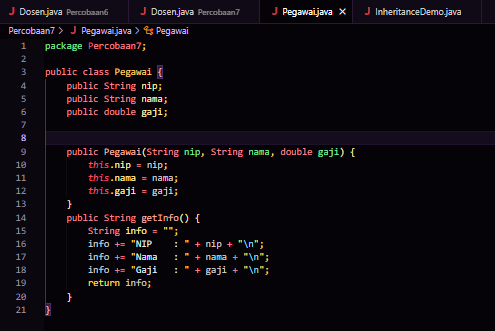


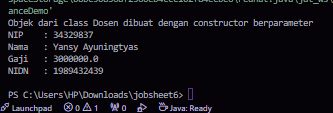




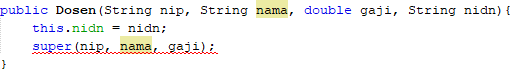
* + 1. Hapus/comment constructor tanpa parameter dari class Pegawai. Tambahkan constructor baru untuk class Pegawai sebagai berikut. Run program kemudian amati hasilnya.

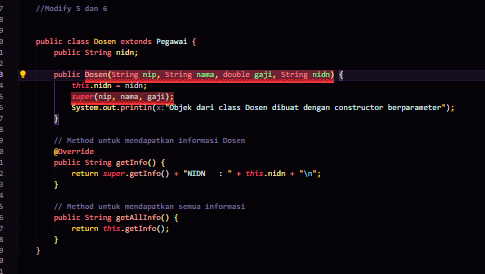


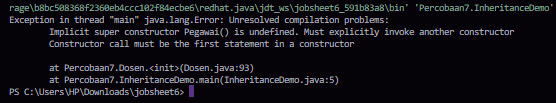




* + 1. Modifikasi constructor pada class Dosen sebagai berikut. Run program kemudian amati hasilnya.

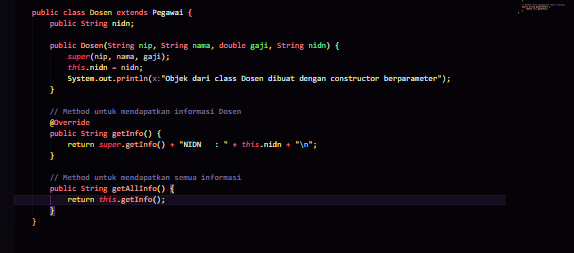


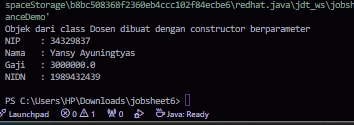




* + 1. Modifikasi constructor pada class Dosen sebagai berikut. Run program kemudian amati hasilnya.





­

## PERTANYAAN

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

**Tidak ada, hasil Ketika di tampilkan sama**

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

**Ya ada, perbedaan terjadi Ketika Langkah 2 bisa menampilkan hasil, sedangkan Langkah 3 mengalami eror** karena The constructor Pegawai() is undefined

* + 1. Mengapa terjadi error pada langkah 4?

**The error in step 4 occurs because you are calling the parameterless constructor of the Employee class implicitly with super(), but that constructor is not defined.**

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

**Constructor Call: In step 3, super() without parameters is invalid if the default constructor is missing from the parent class. In contrast, in step 6, super(nip, nama, gaji) calls the appropriate constructor.**

**Proper Initialization: Step 6 ensures that all parent class attributes are properly initialized, whereas step 3 does not do so effectively.**

* + 1. Mengapa terjadi error pada langkah 5?

**this.nidn = nidn; is now set first to initialize the nidn attribute.**

**super(nip, nama, gaji); is called afterward to call the constructor of the Employee class and initialize the nip, nama, and gaji attributes.**

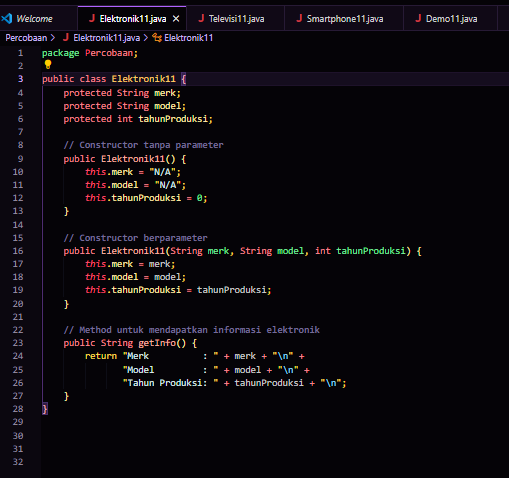
# TUGAS

* 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

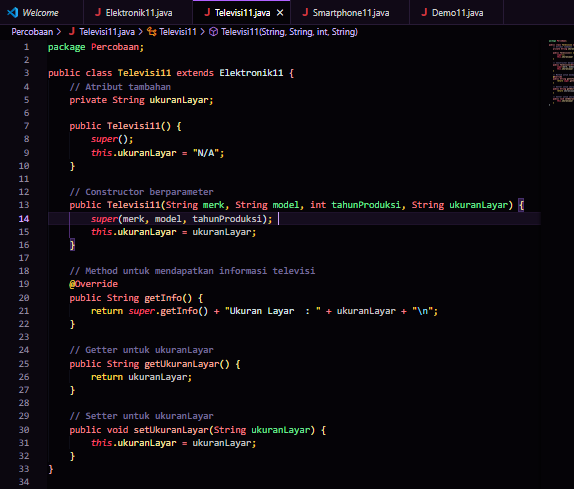
parameter dan constructor berparameter pada masing-masing class. Panggil constructor super() berparameter untuk membuat object dari parent class pada constructor child class.

* 1. Implementasikan class diagram yang dibuat pada mata kuliah PBO teori
  2. Buat class Demo kemudian lakukan instansiasi objek child class pada main function
  3. Cobalah melakukan modifikasi nilai atribut (baik yang dideklarasikan pada child class maupun yang diwariskan darikemudian print info nya.

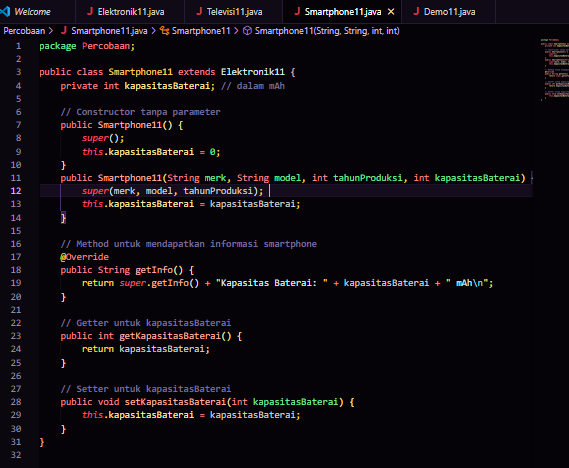
1. Kelas Elektronik



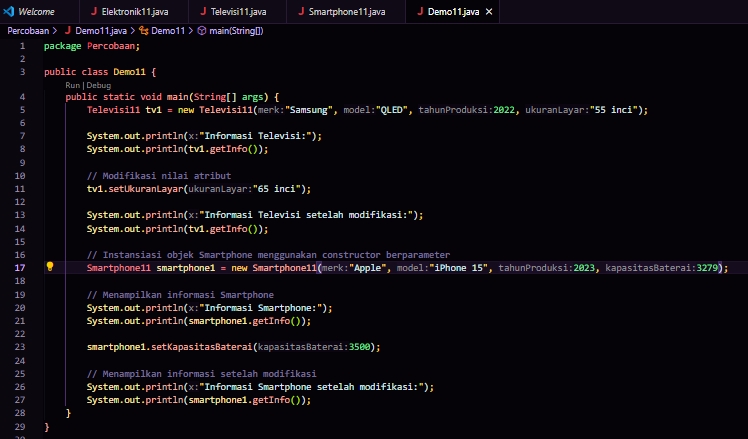
1. Kelas Televisi



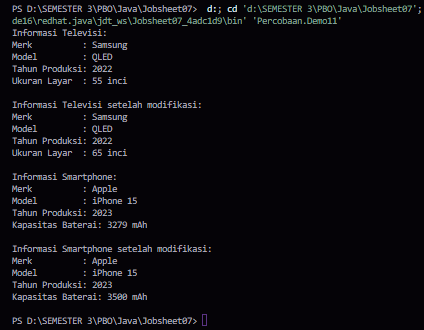
1. Kelas smartphone



1. Kelas demo



1. Hasil Rum



**Getters and Setters: Getters are used to get the value of a private attribute, and setters are used to change the value of that attribute. This follows the principle of encapsulation in object-oriented programming.**

**Using Setters: In the Demo class, we can now use setters to modify the values ​​of the ScreenSize and BatteryCapacity attributes.**

### --- selamat mengerjakan----