**SAFRIZAL RAHMAN 19 SIB 2G**

**LINK**

[**https://github.com/safrizalrahman46/PBO\_SAFRIZ\_THEVIGILANTE**](https://github.com/safrizalrahman46/PBO_SAFRIZ_THEVIGILANTE) **JOBSHEET W06**

**INHERITANCE**

# 1. KOMPETENSI

1. Memahami konsep dasar inheritance atau pewarisan.
2. Mampu membuat suatu subclass dari suatu superclass tertentu.
3. Mampu mengimplementasikan konsep hierarchical inheritance
4. Mampu membuat objek dari suatu subclass dan melakukan pengaksesan terhadap atribut dan method baik yang dimiliki sendiri atau turunan dari superclass nya.

# 2. PENDAHULUAN

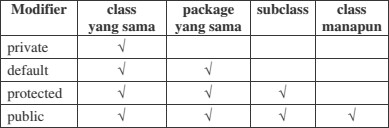
**Inheritance** pada object oriented programming merupakan konsep **pewarisan** dari suatu class yang lebih umum ke suatu class yang lebih spesifik. Kelas yang menurunkan disebut kelas dasar (**base class**/**super class/parent class**), sedangkan kelas yang diturunkan disebut kelas turunan (**derived class**/**sub class**/**child class**). Setiap **subclass** akan “mewarisi” atribut dan method dari **superclass** yang bersifat *public* ataupun *protected*. Manfaat pewarisan adalah *reusability* atau penggunaan kembali baris kode.

Pada bahasa pemrograman Java, deklarasi inheritance dilakukan dengan cara menambahkan kata kunci **extends** setelah deklarasi nama class, kemudian diikuti dengan nama parent class--‐nya. Kata kunci extends tersebut memberitahu kompiler Java bahwa kita ingin melakukan **extension/ perluasan** class. Berikut adalah contoh deklarasi inheritance.

|  |
| --- |
| public class B extends A {  ...  } |

Contoh diatas memberitahukan kompiler Java bahwa class B meng--‐**extend** class A. Artinya, class B adalah subclass dari class A dengan melakukan extension/perluasan. Extension atau perluasan ini akan dilakukan dengan 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

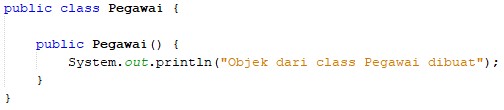
* **super.<namaMethod>()**

Memanggil method parent

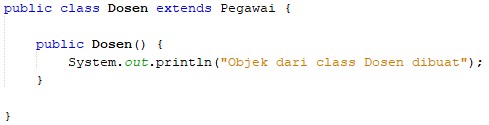
## 1. PERCOBAAN 1 (extends)

**A. TAHAPAN PERCOBAAN**

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

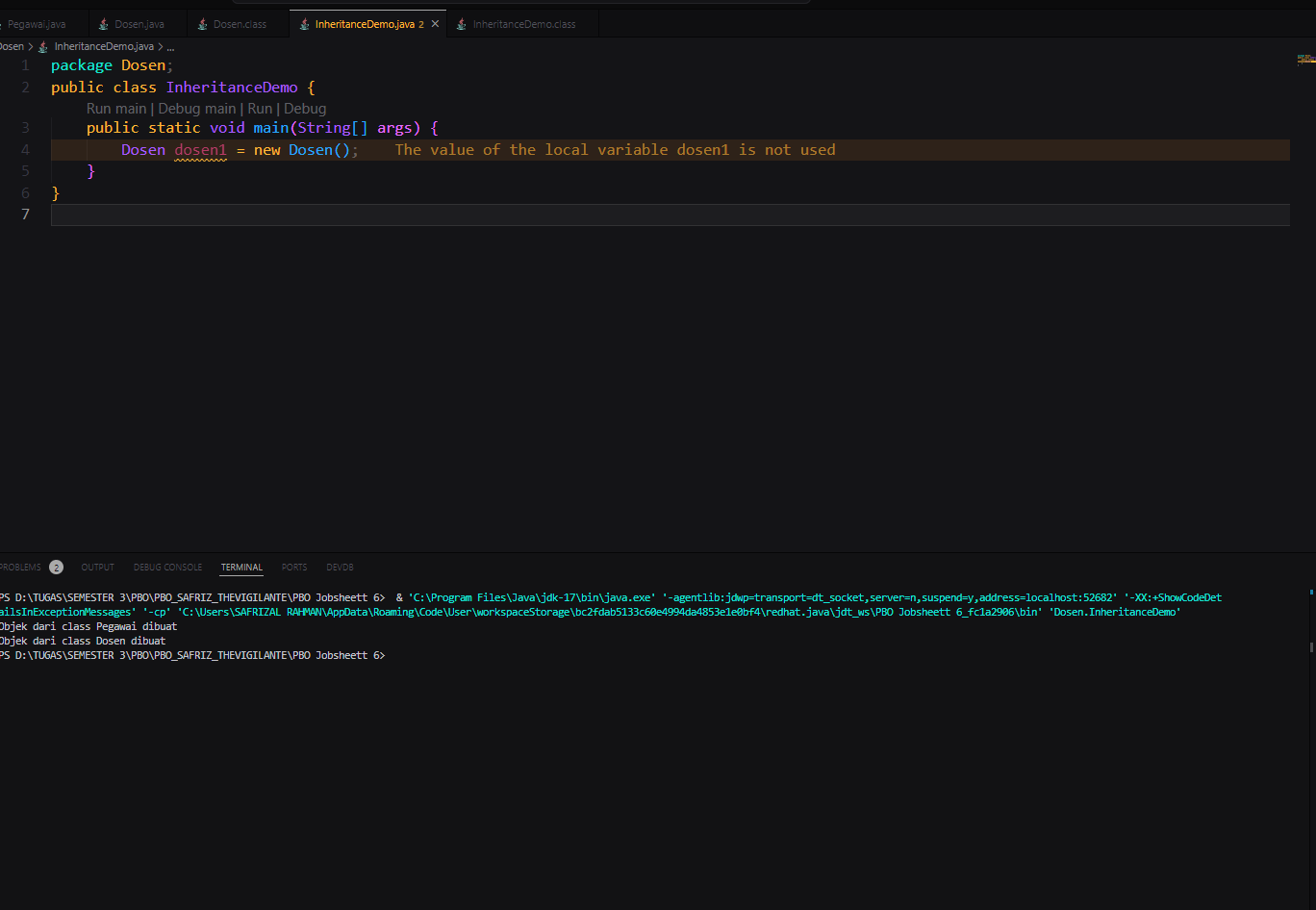


1. Buatlah subclass dari class Pegawai dengan nama Dosen, kemudian buat juga constructor tanpa parameter dengan baris kode berikut:



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





1. Run programnya kemudian amati hasilnya.

## B. PERTANYAAN

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

 **Child class**: Dosen

 **Parent class**: Pegawai

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

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

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

Ada **dua** constructor yang dieksekusi:

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

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

# 4. PERCOBAAN 2 (Pewarisan)

## A. TAHAPAN PERCOBAAN

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

2.

Tambahkan pula atribut NIDN pada class Dosen

3.

Pada

class

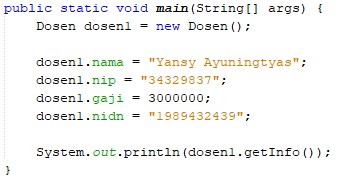
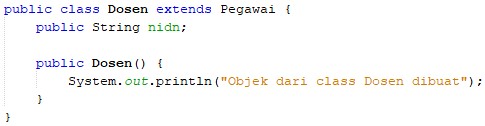
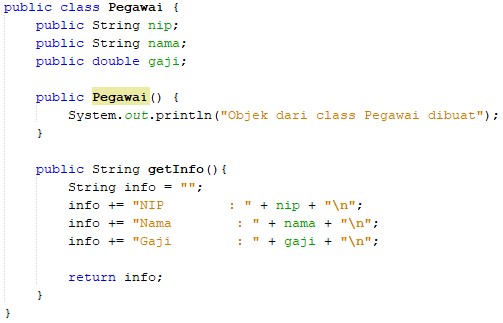
InheritanceDemo

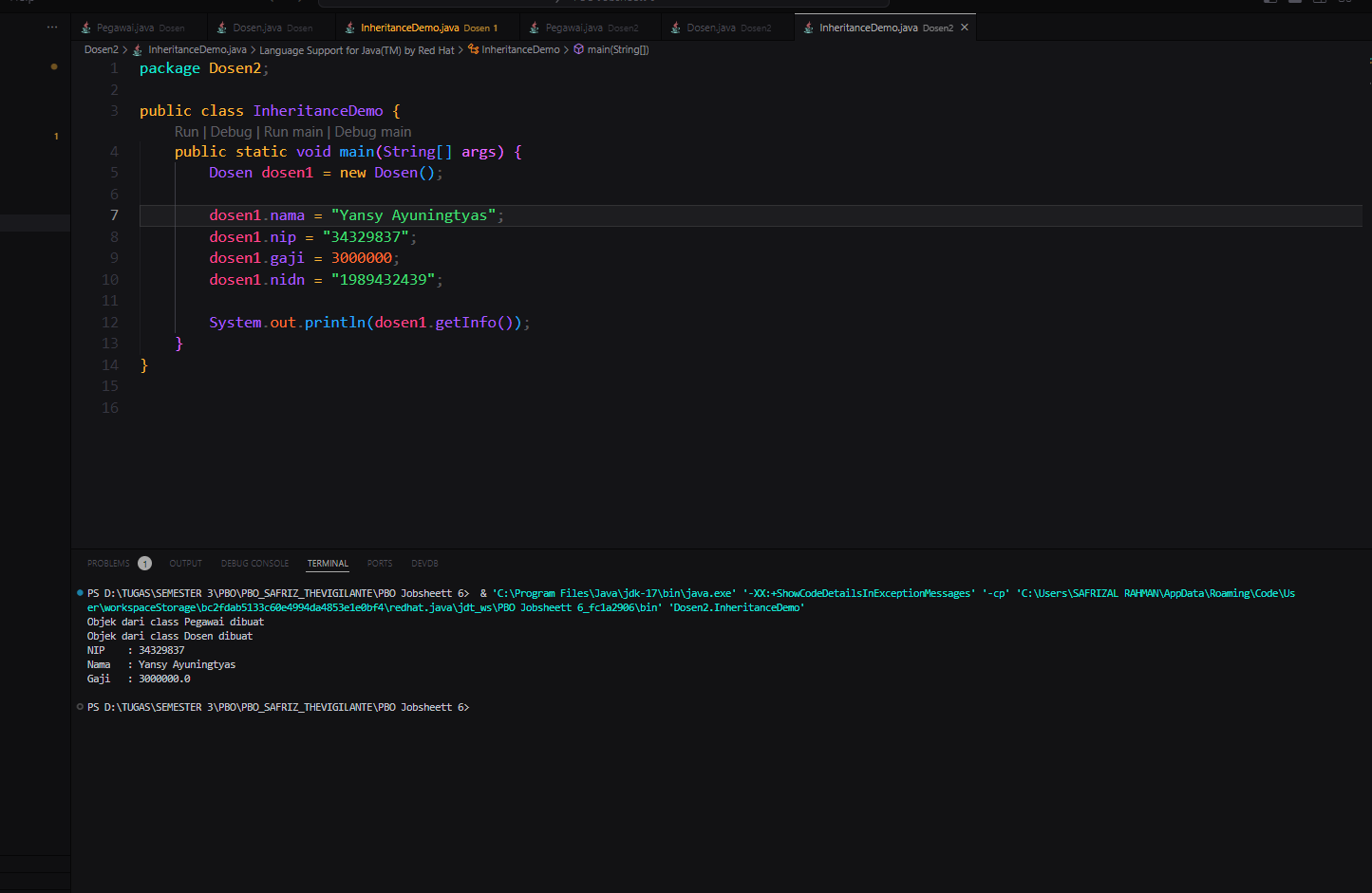
.java

tuliskan baris kode berikut:

4.

Run program kemudian amati hasilnya





## B. PERTANYAAN

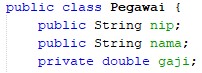
1. Pada percobaan 2 diatas, apakah program dapat berhasil dijalankan ataukah terjadi error?   
   Program **berhasil dijalankan** tanpa error, karena class Dosen secara otomatis mewarisi semua atribut dan method yang ada di class Pegawai.
2. Jika program berhasil dijalankan, mengapa tidak terjadi error pada assignment/pengisian nilai atribut nip, gaji, dan NIDN pada object dosen1 padahal tidak ada deklarasi ketiga atribut tersebut pada class Dosen?   
   Tidak terjadi error karena atribut nip, nama, dan gaji dideklarasikan di class Pegawai, dan class Dosen mewarisi semua atribut dari class Pegawai menggunakan keyword extends. Dengan kata lain, atribut tersebut tersedia dalam objek dosen1, yang merupakan instance dari class Dosen, tetapi atribut tersebut diwarisi dari class Pegawai.
3. Jika program berhasil dijalankan, mengapa tidak terjadi error pada pemanggilan method getInfo() oleh object dosen1 padahal tidak ada deklarasi method getInfo() pada class Dosen?

Tidak terjadi error karena method getInfo() dideklarasikan di class Pegawai, dan class Dosen mewarisi method tersebut. Karena Dosen merupakan subclass dari Pegawai, semua method public atau protected dari class Pegawai tersedia untuk objek dosen1, sehingga getInfo() dapat dipanggil dari objek dosen1 tanpa masalah.

# 5. PERCOBAAN 3 (Hak akses)

## A. TAHAPAN PERCOBAAN

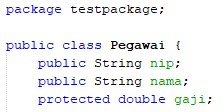
1. Modifikasi access level modifier pada atribut gaji menjadi private pada class Pegawai.java



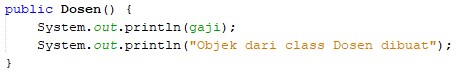
1. Run program kemudian amati hasilnya.
2. Ubah access level modifier atribut gaji menjadi protected kemudian pindah class Pegawai ke package baru, misalnya “testpackage”.

4.

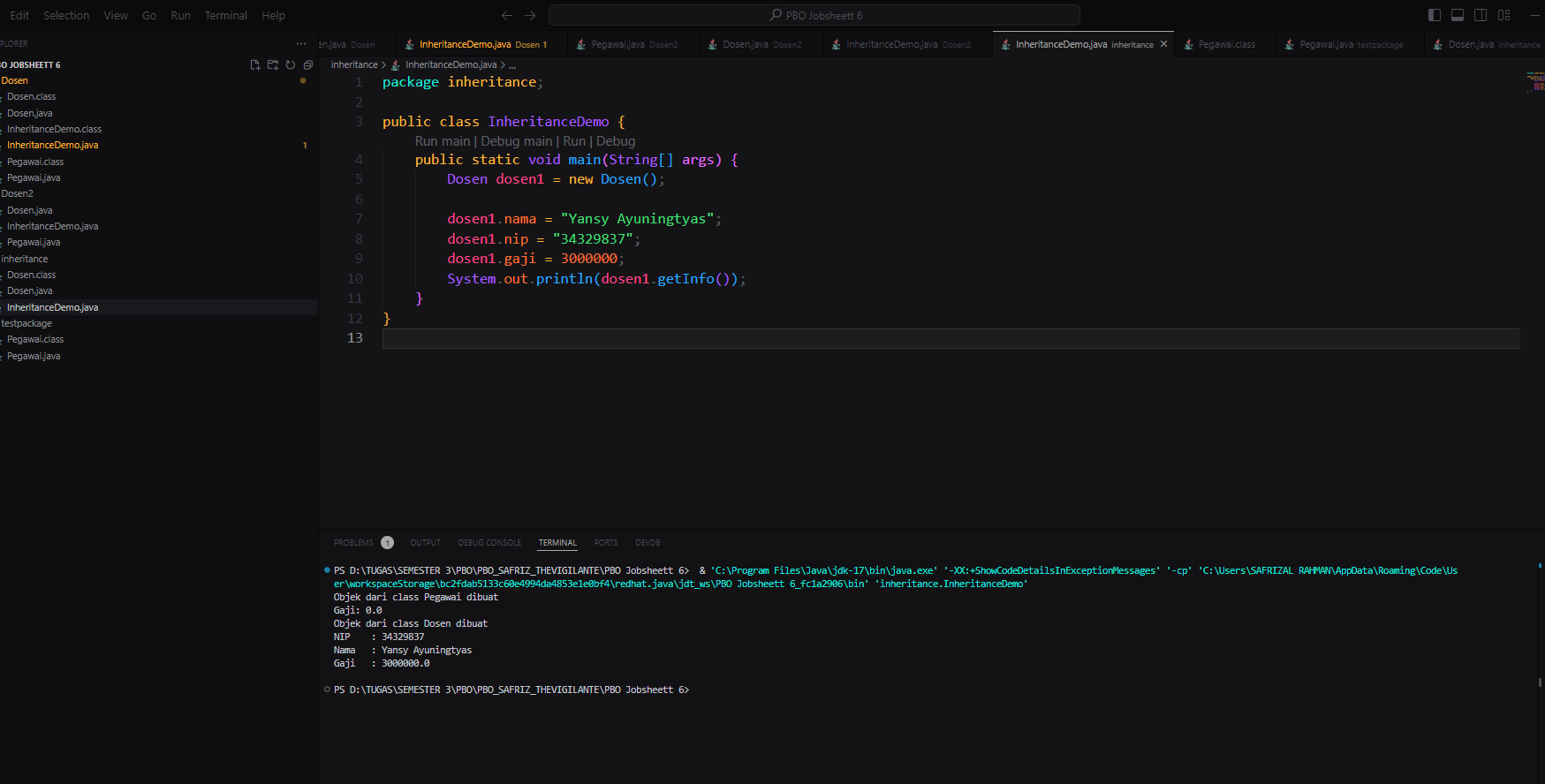
Import class Pegawai dari testpackage pada class Dosen.



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.



## B. PERTANYAAN

1. Pada langkah 1 di atas, terjadi error karena object dosen1 tidak dapat mengakses atribut gaji. Padahal gaji merupakan atribut Pegawai yang merupakan parent class dari Dosen. Mengapa hal ini dapat terjadi?   
   Hal ini terjadi karena akses modifier **private** membatasi akses atribut gaji hanya di dalam class Pegawai itu sendiri. Meskipun Dosen adalah subclass dari Pegawai, atribut yang bersifat **private** tidak dapat diwariskan atau diakses langsung oleh subclass. Hanya metode dari class Pegawai yang dapat mengakses atribut gaji (misalnya melalui getter).
2. Pada langkah 5, setelah class Pegawai berpindah ke package yang berbeda, class Dosen masih dapat mengakses atribut gaji. Mengapa?   
   Setelah access modifier diubah menjadi **protected**, atribut gaji menjadi dapat diakses oleh class yang merupakan subclass, meskipun Dosen berada di package yang berbeda. Modifier **protected** memungkinkan akses ke atribut di subclass, baik di dalam package yang sama maupun package yang berbeda, selama ada hubungan pewarisan.
3. Berdasarkan percobaan tersebut, bagaimana menentukan atribut dan method yang akan diwariskan oleh parent class ke child class?   
   1. **Public**: Atribut dan method dengan modifier **public** dapat diakses dari mana saja, termasuk dari subclass dan package lain.
   2. **Protected**: Atribut dan method dengan modifier **protected** dapat diakses oleh subclass, meskipun berada di package yang berbeda, tetapi tidak oleh class di luar package yang tidak memiliki hubungan pewarisan.
   3. **Default/Package-private**: Jika tidak ada modifier yang diberikan (akses default), atribut dan method hanya bisa diakses oleh class lain dalam package yang sama.
   4. **Private**: Atribut dan method dengan modifier **private** hanya bisa diakses dalam class yang mendeklarasikannya dan tidak bisa diwariskan atau diakses oleh subclass.

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

# 6. PERCOBAAN 4 (Super - atribut)

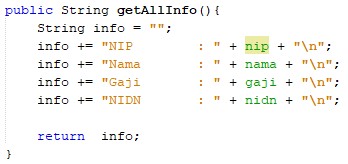
**A.**

**TAHAPAN**

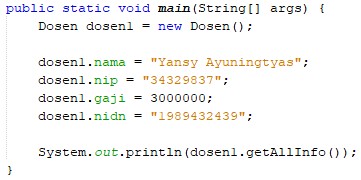
**PERCOBAAN**

1.

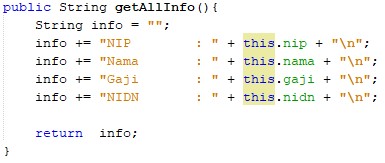
Butlah method getAllInfo() pada class Dosen



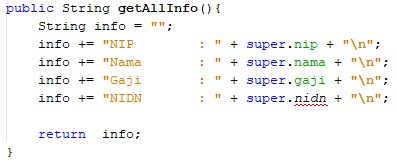
1. Lakukan pemanggilan method getAllInfo() oleh object dosen1 pada class InheritanceDemo.java



1. Run program kemudian amati hasilnya
2. Lakukan modifikasi method getAllInfo() pada class Dosen

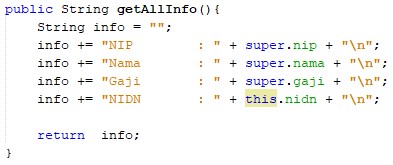


1. Run program kemudian bandingkan hasilnya dengan langkah no 2.
2. Lakukan modifikasi method getAllInfo() pada class Dosen kembali

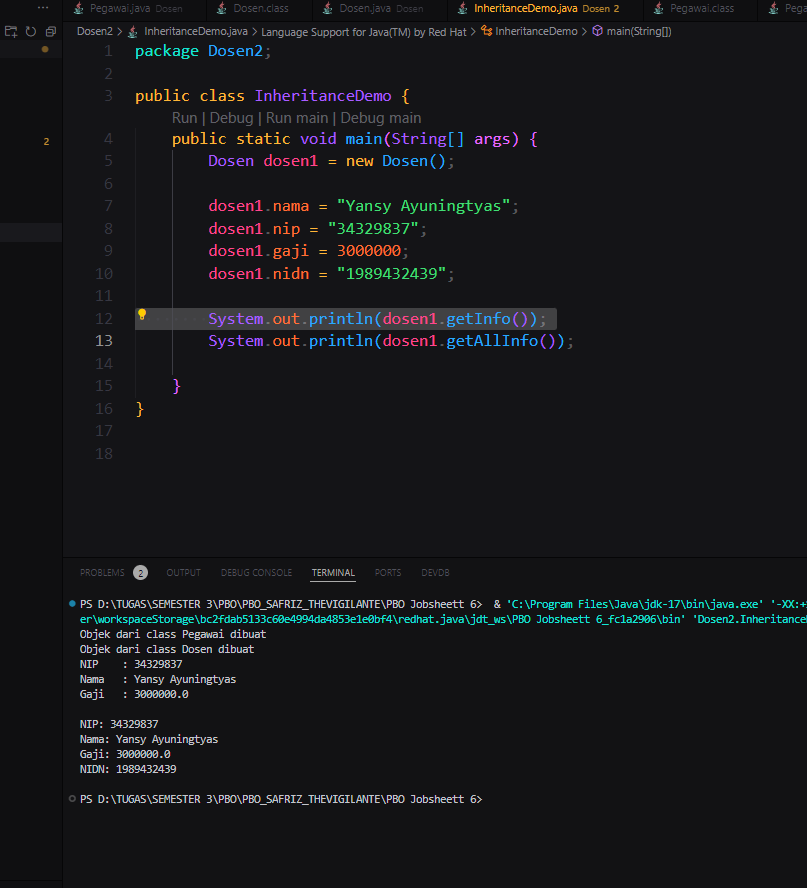


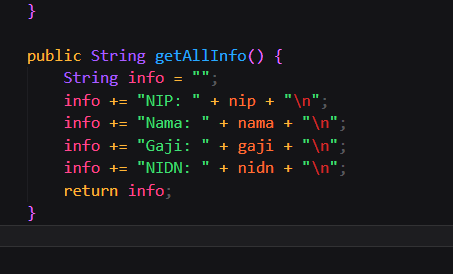
1. Run program kemudian bandingkan hasilnya dengan progam pada no 1 dan no 4.

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



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

SEBELUM THIS  


  
Setelah This

## B. PERTANYAAN

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

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

java

Copy code

info += "NIP: " + nip + "\n";

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

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

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

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

java

Copy code

info += "NIP: " + this.nip + "\n";

info += "Nama: " + this.nama + "\n";

info += "Gaji: " + this.gaji + "\n";

info += "NIDN: " + this.nidn + "\n";

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

java

Copy code

info += "#NIP: \n";

info += "Nama: \n";

info += "Gaji: \n";

info += "NIDN: \n";

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

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

1. Mengapa error terjadi pada program no 6?

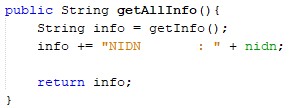
Pada program no 6, Anda mencoba menggunakan super untuk mengakses atribut. Dalam konteks pewarisan (inheritance), super digunakan untuk mengakses atribut atau method dari kelas induk. Jika kelas Dosen tidak mewarisi kelas lain yang memiliki atribut nip, nama, gaji, dan nidn, maka akan terjadi error karena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

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

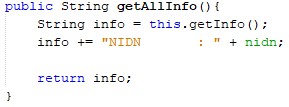
# 7. PERCOBAAN 5 (super & overriding )

## A. TAHAPAN PERCOBAAN

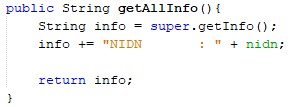
1. Lakukan modifikasi kembali pada method getAllInfo(). Run program kemudian amati hasilnya



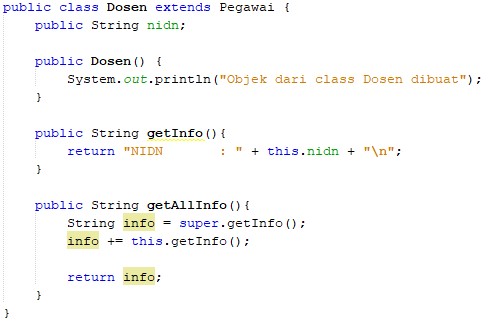
1. Lakukan modifikasi kembali pada method getAllInfo(). Run program kemudian amati hasilnya



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



## B. PERTANYAAN

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

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

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

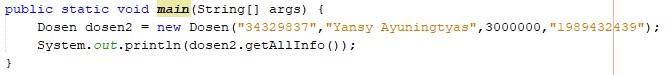
1. Apakah ada perbedaan method super.getInfo() dan this.getInfo() yang dipanggil dalam method getAllInfo() pada langkah 4? Jelaskan!

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

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

1. Pada method apakah terjadi overriding? Jelaskan!

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



# 

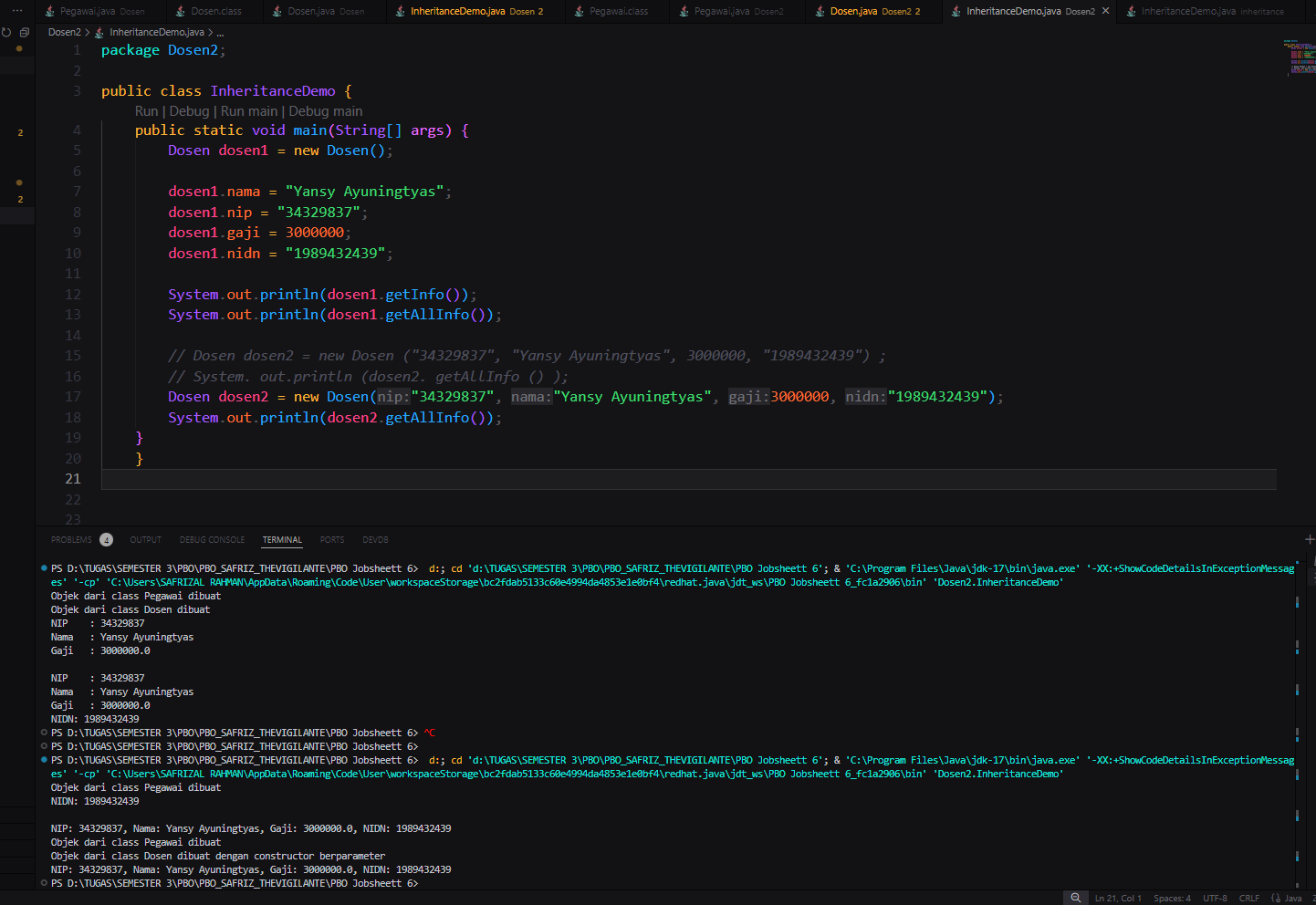
# 8. PERCOBAAN 6 (overloading)

## A. 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.



## B. PERTANYAAN

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

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

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

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

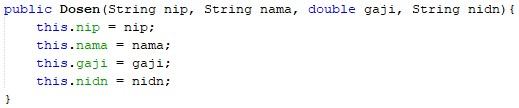
1. Konsep apa dalam OOP yang membolehkan suatu class memiliki constructor atau method dengan nama yang sama dan signature yang berbeda pada satu class?

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

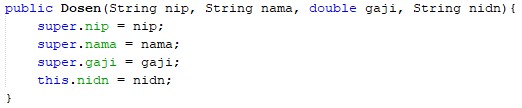
# 9. PERCOBAAN 7 (super - constructor)

## A. TAHAPAN PERCOBAAN

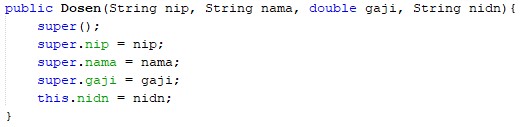
1. Modifikasi constructor pada class Dosen sebagai berikut. Run program kemudian amati hasilnya.



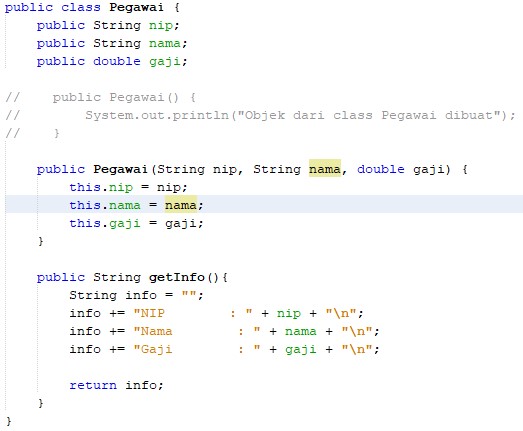
1. Modifikasi constructor pada class Dosen sebagai berikut. Run program kemudian amati hasilnya.



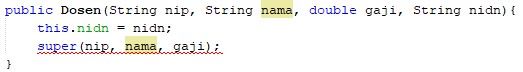
1. Modifikasi constructor pada class Dosen sebagai berikut. Run program kemudian amati hasilnya.



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



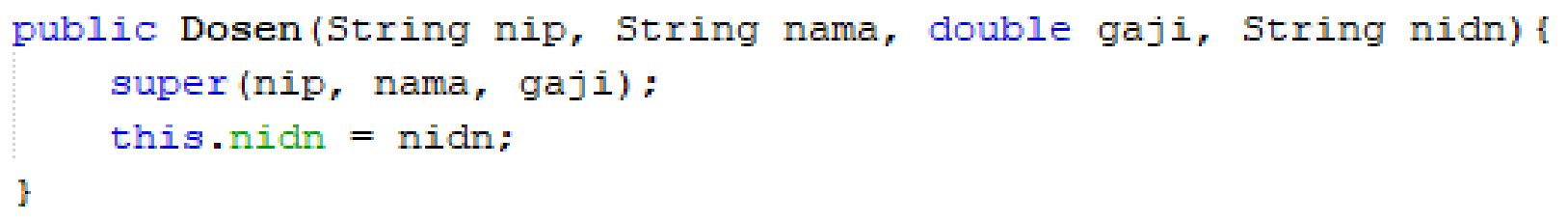
1. Modifikasi constructor pada class Dosen sebagai berikut. Run program kemudian amati hasilnya.



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

**B.**

**PERTANYAAN**



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

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

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

**Difference between Steps 2 and 3:**

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

1. Mengapa terjadi error pada langkah 4?

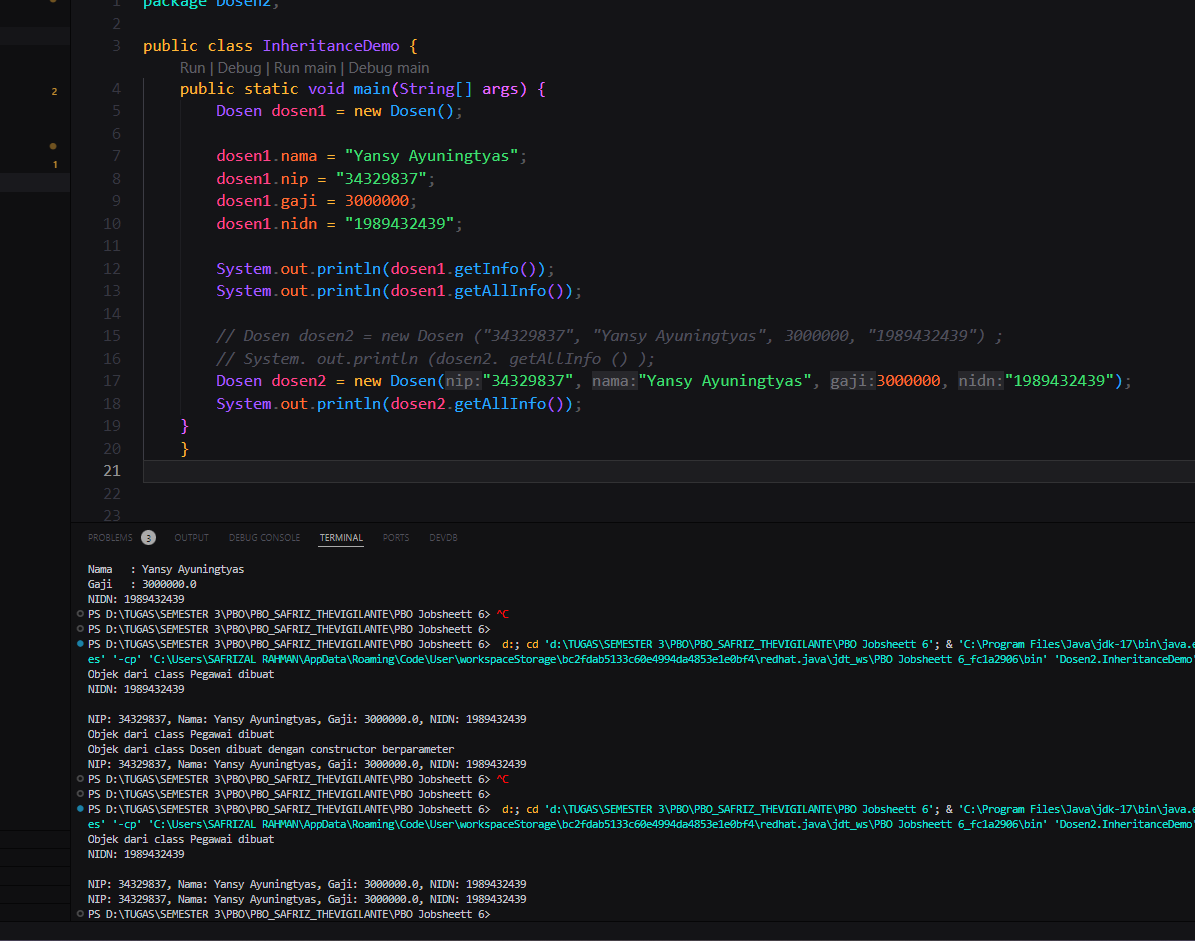
If the default constructor from the Pegawai class is commented out or deleted, you can’t create an instance of Pegawai without parameters. The code tries to instantiate Pegawai with parameters, leading to a compile-time error if no suitable constructor exists.

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

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

1. Mengapa terjadi error pada langkah 5?

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



# 10. TUGAS

1. Tentukan sebuah class yang merupakan turunan dari class yang lain.
2. Buat 3 atribut pada parent class kemudian tambahkan minimal 1 atribut pada child class.
3. Lakukan method overloading dengan membuat 2 constructor yaitu constructor tanpa

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

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.

+------------------------+

| HostingService |

+------------------------+

| - serviceId: String |

| - provider: String |

| - price: double |

+------------------------+

| + HostingService() |

| + HostingService(serviceId, provider, price) |

| + getInfo(): String |

+------------------------+

^

|

+------------------------+

| WebHosting |

+------------------------+

| - domainName: String |

+------------------------+

| + WebHosting() |

| + WebHosting(serviceId, provider, price, domainName) |

| + getInfo(): String |

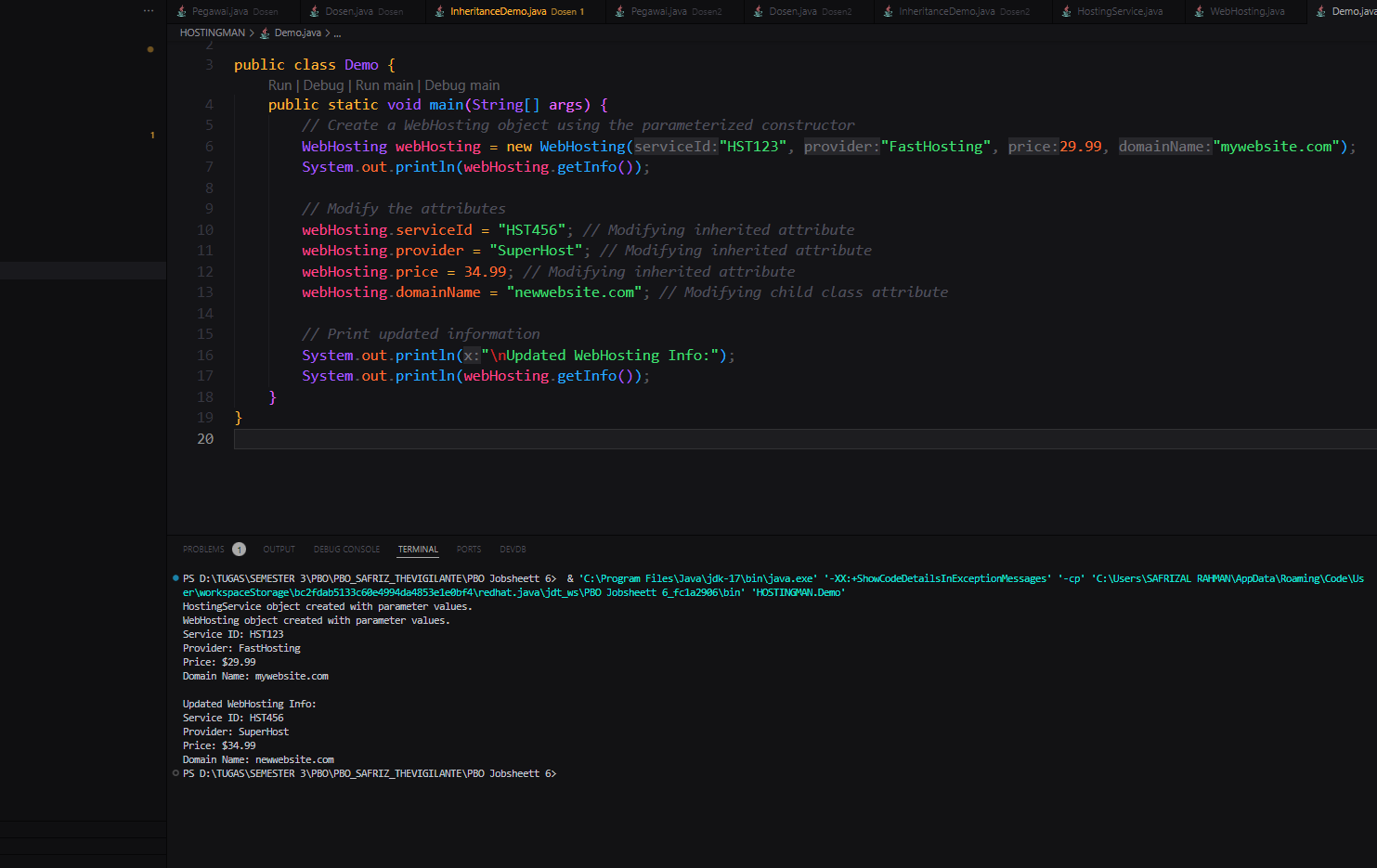
+------------------------+  
  
**6. Explanation of Code Modifications**

* **Instantiation:** In the Demo class, we create an instance of the WebHosting class using the parameterized constructor, passing appropriate values for serviceId, provider, price, and domainName.
* **Modification of Attributes:** We modify the attributes declared in both the parent (HostingService) and child (WebHosting) classes. We can access them directly since they are protected or public.

**Running the Code**

1. **Compile** all the classes.
2. **Run** the Demo class to see the output.

The expected output will display the initial details of the WebHosting instance and then show the updated details after modifying the attributes.



**--- selamat mengerjakan----**