

Jobsheet 7 – Inheritance & Polymorphism
Pemrograman Berbasis Objek



244107060144

Maria Savira

D-IV Sistem Informasi Bisnis / 2C

Percobaan 1 – Extends

1. Membuat class dengan nama pegawai beserta konstruktor tanpa parameter :

```
Pegawai.java > ...  
1 public class Pegawai {  
2     public Pegawai(){  
3         System.out.println(x:"Objek dari class Pegawai dibuat");  
4     }  
5 }
```

2. Membuat subclass dari class Pegawai dengan nama Dosen, juga konstruktor tanpa parameter :

```
6 public class Dosen extends Pegawai {  
7     public Dosen(){  
8         System.out.println(x:"Objek dari class Dosen dibuat");  
9     }  
10 }  
11 }
```

3. Membuat main class dan melakukan instansiasi objek baru bernama dosen1 :

```
1 public class InheritanceDemo {  
    Run | Debug | Run main | Debug main  
2     public static void main(String[] args) {  
3         Pegawai.Dosen dosen1 = new Pegawai.Dosen();  
4     }  
5 }
```

4. Outputnya jika di run :

```
Objek dari class Pegawai dibuat  
Objek dari class Dosen dibuat
```

Pertanyaan

1. Child classnya adalah Dosen, sedangkan parent classnya adalah Pegawai.
2. Kata kunci extends yang berarti “meneruskan” atau “mewariskan” parent Pegawai.
3. Ada 2 konstruktor yang dieksekusi, konstruktor yang dieksekusi terlebih dahulu adalah konstruktor Pegawai. Karena Pegawai merupakan superclassnya.

Percobaan 2 – Pewarisan

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

```
1 public class Pegawai {
2     public String nip;
3     public String nama;
4     public double gaji;
5
6     public Pegawai(){
7         System.out.println(x:"Objek dari class Pegawai dibuat");
8     }
9
10    public static class Dosen extends Pegawai {
11        public Dosen(){
12            System.out.println(x:"Objek dari class Dosen dibuat");
13        }
14    }
15
16    public String getInfo(){
17        String info = "";
18        info += "NIP : " + nip + "\n";
19        info += "Nama : " + nama + "\n";
20        info += "Gaji : " + gaji + "\n";
21
22        return info;
23    }
24 }
```

2. Menambahkan atribut nidn pada class Dosen :

```
public static class Dosen extends Pegawai {
    public String nidn;

    public Dosen(){
        System.out.println(x:"Objek dari class Dosen dibuat");
    }
}
```

3. Menambahkan baris kode dalam InheritanceDemo :

```
1 public class InheritanceDemo {
2     Run | Debug | Run main | Debug main
3     public static void main(String[] args) {
4         Pegawai.Dosen dosen1 = new Pegawai.Dosen();
5
6         dosen1.nama = "Yansy Ayuningtyas";
7         dosen1.nip = "34329837";
8         dosen1.gaji = 3000000;
9         dosen1.nidn = "1989432439";
10
11        System.out.println(dosen1.getInfo());
12    }
}
```

4. Outputnya :

```
Objek dari class Pegawai dibuat
Objek dari class Dosen dibuat
NIP : 34329837
Nama : Yansy Ayuningtyas
Gaji : 3000000.0
```

Pertanyaan

1. Program dapat dijalankan.
2. Karena pengisian nilai beberapa atribut tersebut terjadi dalam parent Pegawai, jadi pengisian atribut tidak perlu dideklarasikan di dalam class Dosen lagi.
3. Karena atribut maupun method secara otomatis diturunkan kepada child Dosen. Jadi tidak akan error.

Percobaan 3 – Hak Akses

1. Mengubah modifier pada atribut gaji menjadi private pada class Pegawai :

```
3 public class Pegawai {
4
5     public String nip;
6     public String nama;
7     private double gaji;
```

2. Output :

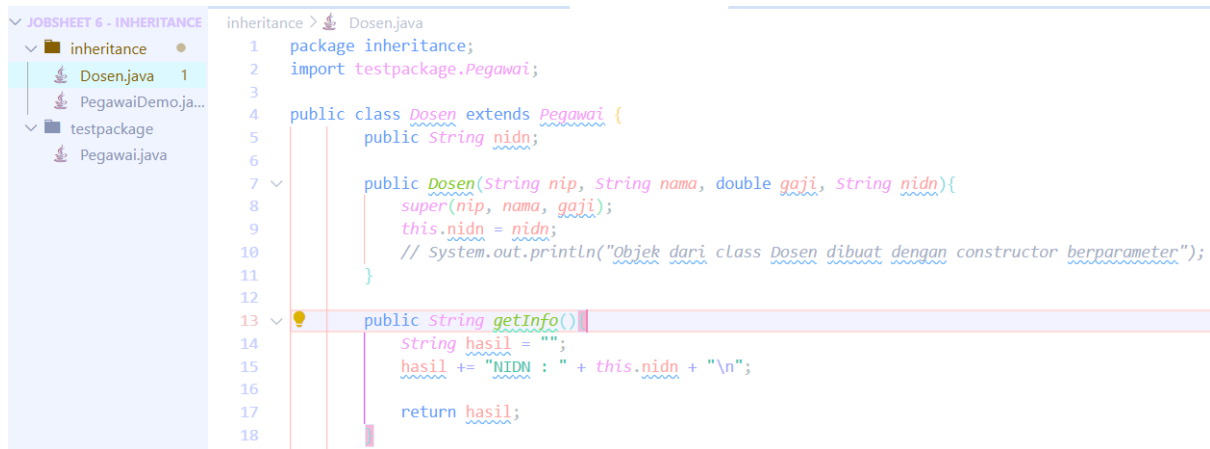
```
heritanceDemo "
Exception in thread "main" java.lang.Error: Unresolved compilation problem:
    The field Pegawai.gaji is not visible

    at InheritanceDemo.main(InheritanceDemo.java:7)
```

3. Memasukkan class Pegawai ke testpackage :

```
1 package testpackage;
2
3 public class Pegawai {
4
5     public String nip;
6     public String nama;
7     protected double gaji;
```

4. Import class Pegawai dari testpackage pada class Dosen :



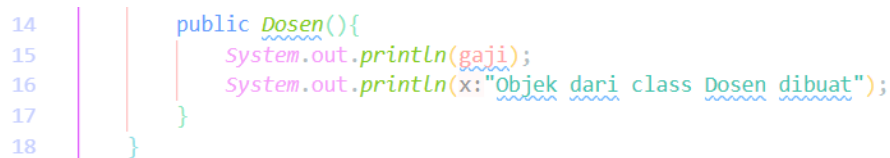
```
1 package inheritance;
2 import testpackage.Pegawai;
3
4 public class Dosen extends Pegawai {
5     public String nidn;
6
7     public Dosen(String nip, String nama, double gaji, String nidn){
8         super(nip, nama, gaji);
9         this.nidn = nidn;
10        // System.out.println("Objek dari class Dosen dibuat dengan constructor berparameter");
11    }
12
13    public String getInfo(){
14        String hasil = "";
15        hasil += "NIDN : " + this.nidn + "\n";
16    }
17    return hasil;
18 }
```

Outputnya :

```
tance_79fa77c7\bin" inheritancee.InheritanceDemo "
Exception in thread "main" java.lang.Error: Unresolved compilation problem:
The field Pegawai.gaji is not visible

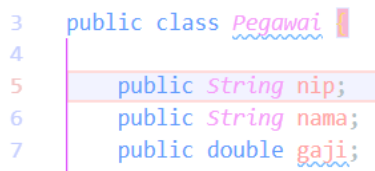
at inheritancee.InheritanceDemo.main(InheritanceDemo.java:10)
```

5. Mencetak atribut gaji pada konstruktor Dosen :



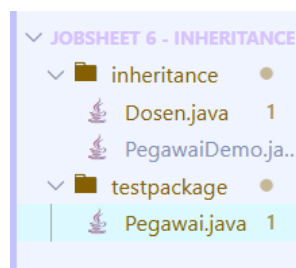
```
14 public Dosen(){
15     System.out.println(gaji);
16     System.out.println(x:"Objek dari class Dosen dibuat");
17 }
18 }
```

Mengembalikan semua modifier ke public :



```
3 public class Pegawai {
4
5     public String nip;
6     public String nama;
7     public double gaji;
8 }
```

6. Letak class Pegawai saat ini :



Pertanyaan

1. Karena atribut gaji diubah modifiernya menjadi private pada class Pegawai. Atribut dengan modifier private hanya bisa diakses langsung di dalam class itu sendiri, tidak bisa diwariskan ke subclass (Dosen) maupun diakses dari luar. Harus melalui setter maupun getter.
2. Jadi meskipun Dosen adalah turunan dari Pegawai, atribut gaji tidak bisa diakses langsung. Karena atribut gaji memiliki modifier protected, atribut dengan modifier protected dapat diakses dari subclass. Karena hal tersebut, class Dosen masih dapat mengakses atribut gaji.
3. Penentuan atribut/method yang diwariskan tergantung pada modifier yang digunakan:
 - Private, berarti tidak diwariskan, hanya bisa diakses di class itu sendiri.
 - Default (tanpa modifier), diwariskan hanya jika masih dalam package yang sama.
 - Protected, diwariskan dan bisa diakses oleh subclass, bahkan jika subclass ada di package berbeda.
 - Public, diwariskan dan bisa diakses di mana saja.

Percobaan 4 – Super Atribut

1. Buat method getAllInfo() pada class Dosen :

```
1  public class Pegawai {
2
3      public String nip;
4      public String nama;
5      public double gaji;
6
7      public Pegawai() {
8          System.out.println(x:"Objek dari class Pegawai dibuat");
9      }
10
11     public String getInfo() {
12         String info = "";
13         info += "NIP : " + nip + "\n";
14         info += "Nama : " + nama + "\n";
15         info += "Gaji : " + gaji + "\n";
16
17         return info;
18     }
19
20     public static class Dosen extends Pegawai {
21         public String nidn;
22
23         public Dosen(){
24             System.out.println(x:"Objek dari class Dosen dibuat");
25         }
26     }
27 }
```

```

27     public String getAllInfo(){
28         String info = "";
29         info += "NIP : " + nip + "\n";
30         info += "Nama : " + nama + "\n";
31         info += "Gaji : " + gaji + "\n";
32         info += "NIDN : " + nidn + "\n";
33
34         return info;
35     }
36 }
37

```

2. Memanggil method getAllInfo() oleh object dosen1 pada class Demo

```

1  public class PegawaiDemo {
2
3      public static void main(String[] args) {
4          Pegawai.Dosen dosen1 = new Pegawai.Dosen();
5
6          dosen1.nama = "Yansy Ayuningtyas";
7          dosen1.nip = "34329837";
8          dosen1.gaji = 3000000;
9          dosen1.nidn = "1989432439";
10
11          System.out.println(dosen1.getAllInfo());
12      }
13 }

```

3. Outputnya ketika di run :

```

Objek dari class Pegawai dibuat
Objek dari class Dosen dibuat
NIP : 34329837
Nama : Yansy Ayuningtyas
Gaji : 3000000.0
NIDN : 1989432439

```

4. Memodifikasi method getAllInfo() dengan menambahkan this :

```

20     public static class Dosen extends Pegawai {
21         public String nidn;
22
23         public Dosen(){
24             System.out.println(x:"Objek dari class Dosen dibuat");
25         }
26
27         public String getAllInfo(){
28             String info = "";
29             info += "NIP : " + this.nip + "\n";
30             info += "Nama : " + this.nama + "\n";
31             info += "Gaji : " + this.gaji + "\n";
32             info += "NIDN : " + this.nidn + "\n";
33
34             return info;
35         }
36     }
37 }

```

5. Output Program nomor 1 :

```
Objek dari class Pegawai dibuat
Objek dari class Dosen dibuat
NIP : 34329837
Nama : Yansy Ayuningtyas
Gaji : 3000000.0
NIDN : 1989432439
```

Output Program nomor 2 :

```
Objek dari class Pegawai dibuat
Objek dari class Dosen dibuat
NIP : 34329837
Nama : Yansy Ayuningtyas
Gaji : 3000000.0
NIDN : 1989432439
```

Outputnya sama. Tidak ada perbedaan

6. Modifikasi getAllInfo() dengan menambahkan super :

```
20     public static class Dosen extends Pegawai {
21         public String nidn;
22
23         public Dosen(){
24             System.out.println(x:"Objek dari class Dosen dibuat");
25         }
26
27         public String getAllInfo(){
28             String info = "";
29             info += "NIP : " + super.nip + "\n";
30             info += "Nama : " + super.nama + "\n";
31             info += "Gaji : " + super.gaji + "\n";
32             info += "NIDN : " + super.nidn + "\n";
33
34             return info;
35         }
36     }
37 }
```

7. Output Program nomor 1 :

```
Objek dari class Pegawai dibuat
Objek dari class Dosen dibuat
NIP : 34329837
Nama : Yansy Ayuningtyas
Gaji : 3000000.0
NIDN : 1989432439
```


Output Program Nomor 3 :

```
Objek dari class Pegawai dibuat
Objek dari class Dosen dibuat
Exception in thread "main" java.lang.Error: Unresolved compilation problem:
    nidn cannot be resolved or is not a field

    at Pegawai$Dosen.getAllInfo(Pegawai.java:32)
    at PegawaiDemo.main(PegawaiDemo.java:10)
```

Perbedaannya adalah output yang pertama berhasil, sedangkan yang kedua gagal karena atribut nidn bukan merupakan atribut parent Pegawai (bukan atribut super), melainkan atribut dari child Dosen.

8. Modifikasi kembali pada method getAllInfo() yaitu mengubah super menjadi this pada pemanggilan atribut nidn :

```
27      public String getAllInfo() {
28          String info = "";
29          info += "NIP : " + super.nip + "\n";
30          info += "Nama : " + super.nama + "\n";
31          info += "Gaji : " + super.gaji + "\n";
32          info += "NIDN : " + this.nidn + "\n";
33
34          return info;
35      }
```

9. Output Program nomor 1 :

```
Objek dari class Pegawai dibuat
Objek dari class Dosen dibuat
NIP : 34329837
Nama : Yansy Ayuningtyas
Gaji : 3000000.0
NIDN : 1989432439
```

Output Program nomor 4 :

```
Objek dari class Pegawai dibuat
Objek dari class Dosen dibuat
NIP : 34329837
Nama : Yansy Ayuningtyas
Gaji : 3000000.0
NIDN : 1989432439
```

Outputnya sekarang berhasil, karena pemanggilan atribut nidn sudah benar. Yaitu menggunakan this, bukan menggunakan super.

Pertanyaan

1. Tidak ada perbedaan pada hasil yang ditampilkan untuk nama, nip, dan gaji. Alasannya karena atribut nama, nip, dan gaji semuanya memang milik parent class (Pegawai), sehingga baik dipanggil dengan this ataupun super, hasilnya tetap sama.

Perbedaan baru muncul jika ada atribut yang tidak dimiliki parent, misalnya nidn (khusus Dosen). Untuk atribut itu, hanya bisa diakses dengan this, tidak bisa dengan super.

2. Error terjadi karena program mencoba mengakses atribut nidn menggunakan super. nidn adalah atribut child class (Dosen), bukan milik parent (Pegawai). Keyword super hanya bisa digunakan untuk mengakses atribut atau method yang berasal dari parent class, bukan atribut yang hanya ada di child.

Jadi, ketika dipanggil super.nidn, compiler tidak mengenali atribut tersebut di parent, sehingga muncul error.

Percobaan 5 – Super & Overriding

1. Modifikasi getAllInfo() untuk mengambil method getInfo() dari parent Pegawai :

```
20     public static class Dosen extends Pegawai {
21         public String nidn;
22
23         public Dosen(){
24             System.out.println(x:"Objek dari class Dosen dibuat");
25         }
26
27         public String getAllInfo(){
28             String info = getInfo();
29             info += "NIDN : " + nidn + "\n";
30
31             return info;
32         }
```

Outputnya :

```
Objek dari class Pegawai dibuat
Objek dari class Dosen dibuat
NIP : 34329837
Nama : Yansy Ayuningtyas
Gaji : 3000000.0
NIDN : 1989432439
```

2. Memodifikasi kembali getAllInfo() dengan menambahkan this pada pemanggilan getInfo() :

```
27 public String getAllInfo(){
28     String info = this.getInfo();
29     info += "NIDN : " + nidn + "\n";
30
31     return info;
32 }
```

Outputnya :

```
Objek dari class Pegawai dibuat
Objek dari class Dosen dibuat
NIP : 34329837
Nama : Yansy Ayuningtyas
Gaji : 3000000.0
NIDN : 1989432439
```

3. Modifikasi menjadi super :

```
27 public String getAllInfo(){
28     String info = super.getInfo();
29     info += "NIDN : " + nidn + "\n";
30
31     return info;
32 }
```

Outputnya :

```
Objek dari class Pegawai dibuat
Objek dari class Dosen dibuat
NIP : 34329837
Nama : Yansy Ayuningtyas
Gaji : 3000000.0
NIDN : 1989432439
```

4. Menambahkan method getInfo() pada class Dosen :

```
27 public String getInfo(){
28     return "NIDN : " + this.nidn + "\n";
29 }
30
31 public String getAllInfo(){
32     String info = super.getInfo();
33     info += getInfo();
}
```

Outputnya :

```
Objek dari class Pegawai dibuat
Objek dari class Dosen dibuat
NIP : 34329837
Nama : Yansy Ayuningtyas
Gaji : 3000000.0
NIDN : 1989432439
```

Pertanyaan

1. Perbedaan getInfo pada masing-masing langkah adalah pada pemanggilan methodnya, untuk praktikum pertama pemanggilannya dilakukan secara biasa (pemanggilan berdasarkan objek tersebut, pada kasus ini objek Dosen), sedangkan pada praktikum kedua dan ketiga berturut-turut memakai this (pemanggilan yang mengacu pada objek dosen sendiri) dan super (pemanggilan dari parent).
2. Bedanya adalah this.getInfo() memanggil objek saat ini. Jika di Dosen sudah di override maka yang dipanggil adalah Dosen.getInfo(). Sedangkan super.getInfo() memanggil getInfo() milik parent Pegawai, dia akan mengabaikan override di Dosen.
3. Overriding terjadi pada method getInfo() karena di parent Pegawai sudah ada method getInfo(), lalu di dalam child Dosen method getInfo() didefinisikan ulang namun dengan implementasi yang beda.
4. Tambahan keyword final pada suatu atribut berarti atribut tersebut merupakan atribut final, tidak bisa diganti, diturunkan kepada subclass, maupun dilakukan override. Program dapat dicompile selama atribut final tidak diberlakukan ketiga pantangan tersebut (di jobsheet tidak ada keterangan untuk menambahkan keyword final bu.. T_T)

Percobaan 6 – Overloading

1. Menambahkan konstruktor baru untuk class Dosen :

```
20 public static class Dosen extends Pegawai {
21     public String nidn;
22
23     public Dosen(String nip, String nama, double gaji, String nidn){
24         System.out.println(x:"Objek dari class Dosen dibuat dengan constructor berparameter");
25     }
```

2. Pada file PegawaiDemo.java

```
public class PegawaiDemo {  
    Run | Debug | Run main | Debug main  
    public static void main(String[] args) {  
        Pegawai.Dosen dosen2 = new Pegawai.Dosen(nip:"34329837", nama:"Yansy Ayuningtyas", gaji:3000000, nidn:"1989432439");  
  
        // dosen1.nama = "Yansy Ayuningtyas";  
        // dosen1.nip = "34329837";  
        // dosen1.gaji = 3000000;  
        // dosen1.nidn = "1989432439";  
  
        System.out.println(dosen2.getAllInfo());  
    }  
}
```

Outputnya :

```
Objek dari class Pegawai dibuat  
Objek dari class Dosen dibuat dengan constructor berparameter  
NIP : null  
Nama : null  
Gaji : 0.0  
NIDN : null
```

Pertanyaan

1. Pada praktikum tersebut, nilai, nip, nama, gaji, dan nidn null karena value dari objek belum disimpan, isinya masih ada sebatas parameter saja. Belum terdapat deklarasi untuk mengisi value objek pada class Dosen.
2. Cara kerjanya berbeda, konstruktor tanpa parameter memiliki signature Dosen() yang tidak menerima argumen. Sedangkan konstruktor yang dibuat pada langkah 1 memiliki parameter (String nip, String nama, double gaji, String nidn). Karena jumlah dan tipe parameternya berbeda, maka signaturenya juga berbeda.
3. Konsep itu disebut Method Overloading (atau Constructor Overloading jika berlaku pada konstruktor). Overloading artinya dalam satu class boleh ada lebih dari satu method/konstruktor dengan nama yang sama, asalkan jumlah atau tipe parameternya berbeda.

Percobaan 7 – Super Constructor

1. Modifikasi konstruktor Dosen dengan memasukkan value yang didapat dari parameter ke dalam atribut Objek.

```
20 public static class Dosen extends Pegawai {
21     public String nidn;
22
23     public Dosen(String nip, String nama, double gaji, String nidn){
24         this.nip = nip;
25         this.nama = nama;
26         this.gaji = gaji;
27         this.nidn = nidn;
28         // System.out.println("Objek dari class Dosen dibuat dengan constructor berparameter");
29     }
```

Output :

```
Objek dari class Pegawai dibuat
NIP : 34329837
Nama : Yansy Ayuningtyas
Gaji : 3000000.0
NIDN : 1989432439
```

2. Modifikasi pada child Dosen :

```
21 public static class Dosen extends Pegawai {
22     public String nidn;
23
24     public Dosen(String nip, String nama, double gaji, String nidn){
25         super.nip = nip;
26         super.nama = nama;
27         super.gaji = gaji;
28         this.nidn = nidn;
29         // System.out.println("Objek dari class Dosen dibuat dengan constructor berparameter");
30     }
```

Output :

```
NIP : 34329837
Nama : Yansy Ayuningtyas
Gaji : 3000000.0
NIDN : 1989432439
```

3. Modifikasi pada child Dosen :

```
24 public Dosen(String nip, String nama, double gaji, String nidn){
25     super();
26     super.nip = nip;
27     super.nama = nama;
28     super.gaji = gaji;
29     this.nidn = nidn;
30     // System.out.println("Objek dari class Dosen dibuat dengan constructor berparameter");
31 }
```

Output :

```
NIP : 34329837
Nama : Yansy Ayuningtyas
Gaji : 3000000.0
NIDN : 1989432439
```

4. Modifikasi konstruktor default pada konstruktor Pegawai menjadi konstruktor berparameter:

```
3 public class Pegawai {
4
5     public String nip;
6     public String nama;
7     public double gaji;
8
9     public Pegawai(String nip, String nama, double gaji) {
10         this.nip = nip;
11         this.nama = nama;
12         this.gaji = gaji;
13     }
14
15     public String getInfo() {
16         String info = "";
17         info += "NIP : " + nip + "\n";
18         info += "Nama : " + nama + "\n";
19         info += "Gaji : " + gaji + "\n";
20     }
```

Output :

```
tasi Objek\Jobsheet 6 - Inheritance" && cmd /C ""C:\Program Files\Java\jdk21.0.1\bin\java.exe" -XX:+ShowCodeDetailsInExceptionMessages -cp "C:\Users\ming\Code\User\workspaceStorage\ad5afc1d5604167b803068a139f649d2\redhat.java\jdt_ws\Jobsheet 6 - Inheritance_79fa77c7\bin" inheritance.PegawaiDemo "
Exception in thread "main" java.lang.Error: Unresolved compilation problem:
    The constructor Pegawai() is undefined

at testpackage.Pegawai$Dosen.<init>(Pegawai.java:28)
at inheritance.PegawaiDemo.main(PegawaiDemo.java:6)
```

5. Modifikasi konstruktor pada child Dosen :

```
24 public static class Dosen extends Pegawai {
25     public String nidn;
26
27     public Dosen(String nip, String nama, double gaji, String nidn) {
28         this.nidn = nidn;
29         super(nip, nama, gaji);
30         // System.out.println("Objek dari class Dosen dibuat dengan constructor berparameter");
31     }
```

Output :

```
Implicit super constructor Pegawai() is undefined. Must explicitly invoke another constructor
Constructor call must be the first statement in a constructor
```

```
at testpackage.Pegawai$Dosen.<init>(Pegawai.java:27)
at inheritance.PegawaiDemo.main(PegawaiDemo.java:6)
```

6. Modifikasi konstruktor child Dosen :

```
24     public static class Dosen extends Pegawai {  
25         public String nidn;  
26  
27         public Dosen(String nip, String nama, double gaji, String nidn){  
28             super(nip, nama, gaji);  
29             this.nidn = nidn;  
30             // System.out.println("Objek dari class Dosen dibuat dengan constructor berparameter");  
31         }  
    }
```

Output :

```
NIP : 34329837  
Nama : Yansy Ayuningtyas  
Gaji : 3000000.0  
NIDN : 1989432439
```

Pertanyaan

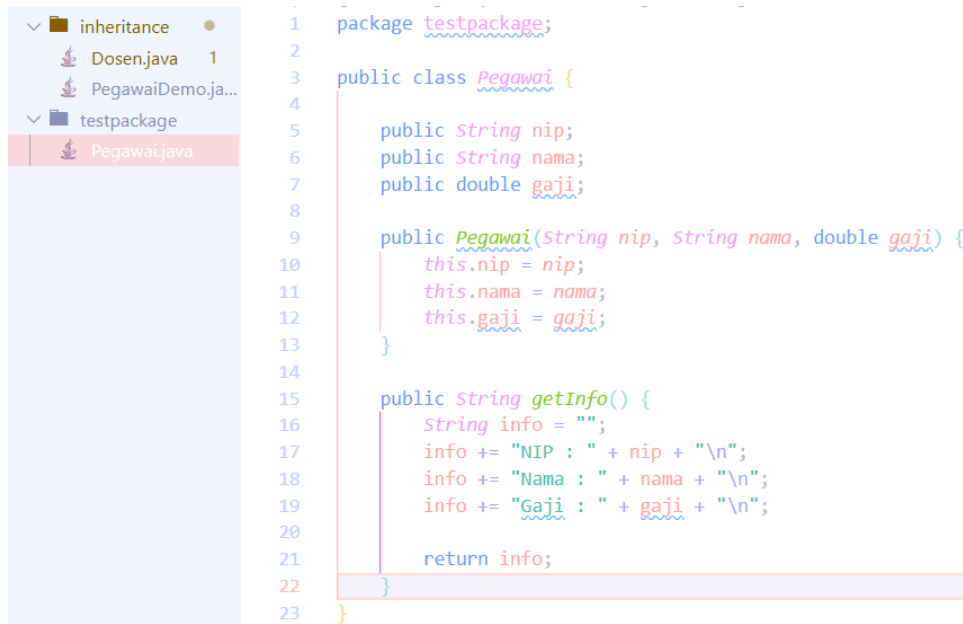
1. Pada langkah 1, menyimpan semua value variabel langsung ke dalam child Dosen (mengacu langsung pada class Dosen).
Sedangkan pada langkah 2, menyimpan value dari variabel nip, nama, dan gaji melalui atribut super atau parent classnya yaitu Pegawai.
2. Perbedaannya adalah langkah 2 menyimpan value dari variabel nip, nama, dan gaji melalui atribut super atau parent classnya yaitu Pegawai.
Pada langkah 3, konstruktor dari superclass dipanggil terlebih dahulu sebelum menyimpan value variabel.
3. Langkah 4 error karena konstruktor default Pegawai diganti dengan konstruktor berparameter, sehingga terdapat error pada class Dosen karena pada konstruktornya belum diganti menyesuaikan konstruktor baru dari Pegawai.
4. Super() yang digunakan pada langkah 3 mengacu pada pemanggilan konstruktor default, sedangkan pada langkah 6 super(nip, nama, gaji) merupakan pengaplikasian dari konstruktor berparameter milik superclass.
5. Pada langkah 5 terjadi error karena konstruktor dari superclass yang memiliki parameter tidak didahulukan, sedangkan alur dari inheritance sendiri berawal dari superclass.

Tugas Tambahan

Menambahkan atribut baru (bebas) pada child Dosen dan menambahkan 1 object Dosen baru.

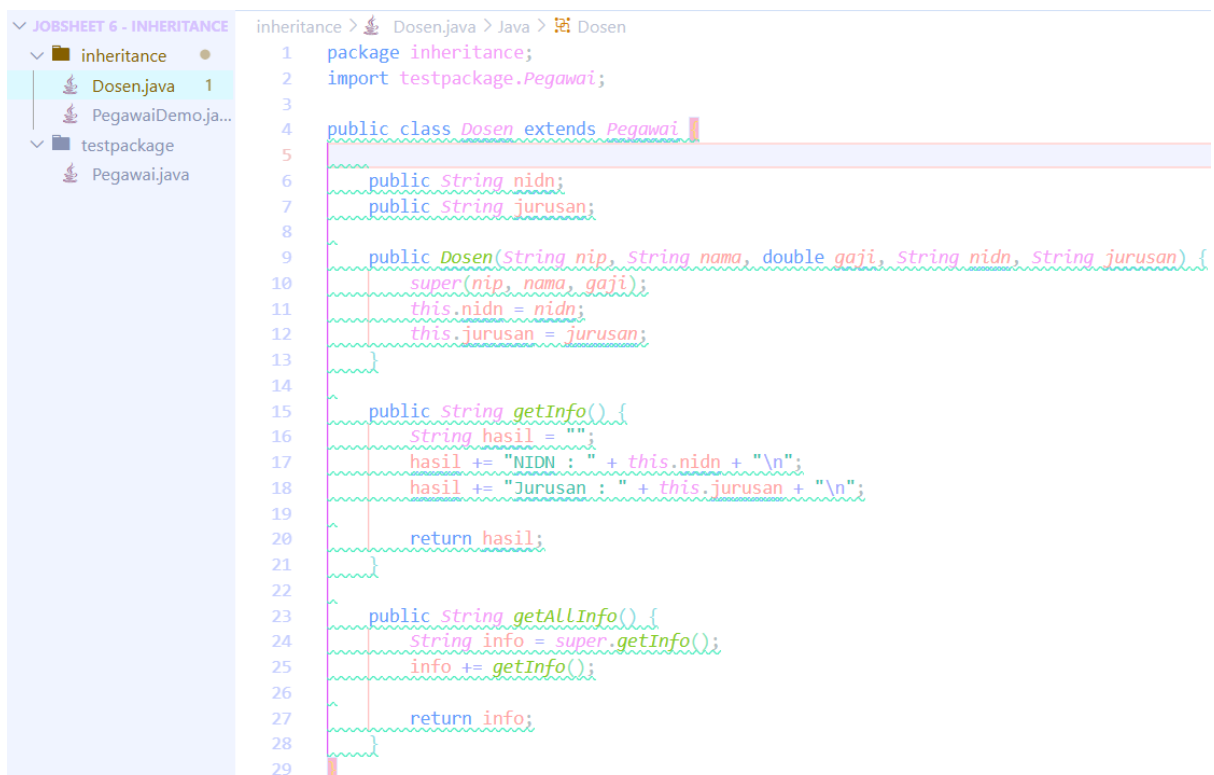
Kode Program

Class Pegawai



```
1 package testpackage;
2
3 public class Pegawai {
4
5     public String nip;
6     public String nama;
7     public double gaji;
8
9     public Pegawai(String nip, String nama, double gaji) {
10         this.nip = nip;
11         this.nama = nama;
12         this.gaji = gaji;
13     }
14
15     public String getInfo() {
16         String info = "";
17         info += "NIP : " + nip + "\n";
18         info += "Nama : " + nama + "\n";
19         info += "Gaji : " + gaji + "\n";
20
21         return info;
22     }
23 }
```

Class Dosen



```
inheritance > Dosen.java > Java > Dosen
1 package inheritance;
2 import testpackage.Pegawai;
3
4 public class Dosen extends Pegawai {
5
6     public String nidn;
7     public String jurusan;
8
9     public Dosen(String nip, String nama, double gaji, String nidn, String jurusan) {
10         super(nip, nama, gaji);
11         this.nidn = nidn;
12         this.jurusan = jurusan;
13     }
14
15     public String getInfo() {
16         String hasil = "";
17         hasil += "NIDN : " + this.nidn + "\n";
18         hasil += "Jurusan : " + this.jurusan + "\n";
19
20         return hasil;
21     }
22
23     public String getAllInfo() {
24         String info = super.getInfo();
25         info += getInfo();
26
27         return info;
28     }
29 }
```

PegawaiDemo



```
1 package inheritance;
2
3 public class PegawaiDemo {
4     public static void main(String[] args) {
5         Dosen dosen2 = new Dosen(nip:"34329837", nama:"Yansy Ayuningtyas", gaji:3000000, nidn:"1989432439", jurusan:"Akuntansi");
6         Dosen dosen3 = new Dosen(nip:"12345678", nama:"Bu Laras", gaji:3000000, nidn:"0987654321", jurusan:"Teknologi Informasi");
7
8         // dosen1.nama = "Yansy Ayuningtyas";
9         // dosen1.nip = "34329837";
10        // dosen1.gaji = 3000000;
11        // dosen1.nidn = "1989432439";
12
13        System.out.println(dosen2.getAllInfo());
14        System.out.println(dosen3.getAllInfo());
15    }
16 }
```

Output

```
NIP : 34329837
Nama : Yansy Ayuningtyas
Gaji : 3000000.0
NIDN : 1989432439
Jurusan : Akuntansi

NIP : 12345678
Nama : Bu Laras
Gaji : 3000000.0
NIDN : 0987654321
Jurusan : Teknologi Informasi
```

Tugas

1. Class turunan : Lagu
Superclass : Penyanyi

```
Welcome  Penyanyi.java  PenyanyiDemo.java
tugas > Penyanyi.java > Java > Penyanyi
1  package tugas;
2
3  class Penyanyi {
4      String nama;
5      String genre;
6      int tahunDebut;
7
8      public Penyanyi() {
9      }
10
11     public static class Lagu extends Penyanyi {
12         String judulLagu;
13
14         public Lagu() {
15             super();
16         }
17
18         public Lagu(String nama, String genre, int tahunDebut, String judulLagu) {
19             super(nama, genre, tahunDebut);
20             this.judulLagu = judulLagu;
21         }
22
23         @Override
24         public void info() {
25             System.out.println("=== Data Lagu ===");
26             System.out.println("Penyanyi      : " + nama);
27             System.out.println("Genre       : " + genre);
28             System.out.println("Tahun Debut : " + tahunDebut);
29             System.out.println("Judul Lagu  : " + judulLagu);
30         }
31     }
32 }
```

2. 3 atribut pada parent class :

```
3  class Penyanyi {
4      String nama;
5      String genre;
6      int tahunDebut;
```

Menambahkan 1 atribut pada child class :

```
11     public static class Lagu extends Penyanyi {
12         String judulLagu;
```

3. Konstruktor tanpa parameter dan berparameter pada class Penyanyi (overloading pada konstruktor tanpa parameter) :

```
8      public Penyanyi() {
9      }
10
11 >   public static class Lagu extends Penyanyi { ...
33
34      public Penyanyi(String nama, String genre, int tahunDebut) {
35          this.nama = nama;
36          this.genre = genre;
37          this.tahunDebut = tahunDebut;
38      }
```

Konstruktor tanpa parameter dan berparameter pada class Lagu (overloading pada konstruktor tanpa parameter) :

```
11      public static class Lagu extends Penyanyi {
12          String judulLagu;
13
14          public Lagu() {
15              super();
16          }
17
18          public Lagu(String nama, String genre, int tahunDebut, String judulLagu) {
19              super(nama, genre, tahunDebut);
20              this.judulLagu = judulLagu;
21          }
22      }
```

4. Override pada method dalam parent class dan child class :

```
23      @Override
24      public void info() {
25          System.out.println(x:"=== Data Lagu ===");
26          System.out.println("Penyanyi      : " + nama);
27          System.out.println("Genre       : " + genre);
28          System.out.println("Tahun Debut : " + tahunDebut);
29          System.out.println("Judul Lagu  : " + judulLagu);
30      }
31  }
32
33 >   public Penyanyi(String nama, String genre, int tahunDebut) { ...
38
39      public void info() {
40          System.out.println(x:"=== Data Penyanyi ===");
41          System.out.println("Nama       : " + nama);
42          System.out.println("Genre      : " + genre);
43          System.out.println("Tahun Debut : " + tahunDebut);
44      }
```

Pada line 23-31 merupakan method dalam child Lagu. Sedangkan line 39-44 merupakan method dalam parent Penyanyi.

5. Instansiasi objek child Lagu dalam PenyanyiDemo :



```
1 package tugas;
2
3 public class PenyanyiDemo {
4     public static void main(String[] args) {
5         Penyanyi.Lagu lagu1 = new Penyanyi.Lagu(nama:"Taylor Swift", genre:"Pop", tahunDebut:2006, judulLagu:"Love Story");
6         Penyanyi.Lagu lagu2 = new Penyanyi.Lagu();
7         lagu2.nama = "Dewa 19";
8         lagu2.genre = "Pop Rock";
9         lagu2.tahunDebut = 1986;
10        lagu2.judulLagu = "Aku Milikmu";
11
12        lagu1.info();
13        System.out.println();
14        lagu2.info();
15    }
16 }
```

Output :

```
=== Data Lagu ===
Penyanyi    : Taylor Swift
Genre       : Pop
Tahun Debut : 2006
Judul Lagu  : Love Story

=== Data Lagu ===
Penyanyi    : Dewa 19
Genre       : Pop Rock
Tahun Debut : 1986
Judul Lagu  : Aku Milikmu
```

Link GitHub :

[Semester-3/Pemrograman Berorientasi Objek/Jobsheet 7 - Inheritance dan Polimorfisme at main · MariaSavira/Semester-3](#)