

JOBSHEET W07

INHERITANCE & POLYMORPHISM

1. 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

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.

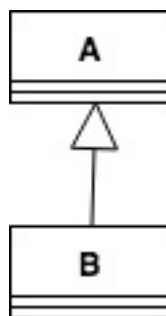
```
public class B extends A {  
    ...  
}
```

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 penambahan 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.

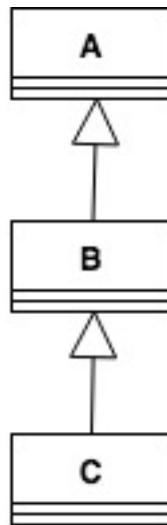


Gambar 1. Contoh Single Inheritance

2. Multilevel Inheritance

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

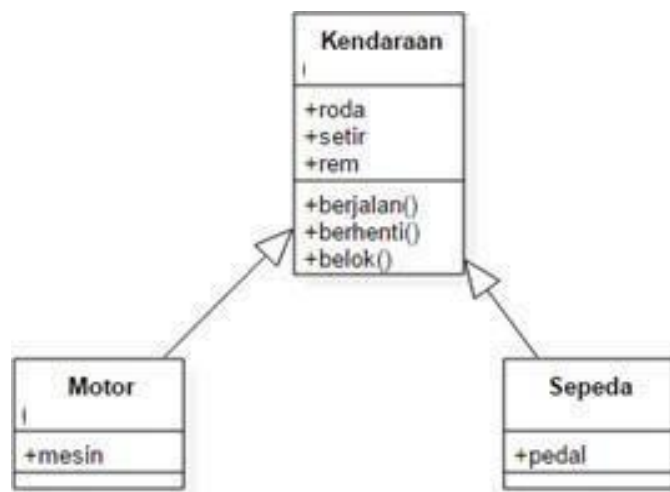
Contoh:



Gambar 2. Contoh Multilevel Inheritance

Pada Gambar 2 di atas dapat dilihat bahwa class B merupakan subclass dari class A, namun dia juga merupakan 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:

| Modifier | class yang sama | package yang sama | subclass | class manapun |
|-----------|-----------------|-------------------|----------|---------------|
| private | √ | | | |
| default | √ | √ | | |
| protected | √ | √ | √ | |
| public | √ | √ | √ | √ |

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()**
Memanggil 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 dalam constructor 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:

1. Overloading

- Method overloading berarti kondisi dimana ada method dengan nama yang sama, tetapi memiliki 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 dalam hierarki pewarisan.
- Karakteristik overloading: nama method sama, method signature berbeda, return type boleh sama atau berbeda.
- JVM menentukan method mana yang akan dipanggil pada compile-time 2 compile-time polymorphism
- Disebut juga static binding atau early binding

2. 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

3. PERCOBAAN 1 (extends)

A. TAHAPAN PERCOBAAN

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

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

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

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

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

```
public static void main(String[] args) {  
    Dosen dosen1 = new Dosen();  
}
```

4. Run programnya kemudian amati hasilnya.

```
InheritanceDemo'  
Objek dari class Pegawai dibuat  
Objek dari class Dosen dibuat
```

B. PERTANYAAN

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

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

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

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

```
public class Dosen extends Pegawai {
```

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

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

4. PERCOBAAN 2 (Pewarisan)

A. TAHAPAN PERCOBAAN

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

```
public class Pegawai {  
    public String nip;  
    public String nama;  
    public double gaji;  
  
    public Pegawai() {  
        System.out.println("Objek dari class Pegawai dibuat");  
    }  
  
    public String getInfo() {  
        String info = "";  
        info += "NIP          : " + nip + "\n";  
        info += "Nama          : " + nama + "\n";  
        info += "Gaji          : " + gaji + "\n";  
  
        return info;  
    }  
}
```

2. Tambahkan pula atribut NIDN pada class Dosen

```
public class Dosen extends Pegawai {  
    public String nidn;  
  
    public Dosen() {  
        System.out.println("Objek dari class Dosen dibuat");  
    }  
}
```

3. Pada class InheritanceDemo.java tuliskan baris kode berikut:

```
public static void main(String[] args) {  
    Dosen dosen1 = new Dosen();  
  
    dosen1.nama = "Yansy Ayuningtyas";  
    dosen1.nip = "34329837";  
    dosen1.gaji = 3000000;  
    dosen1.nidn = "1989432439";  
  
    System.out.println(dosen1.getInfo());  
}
```

4. Run program kemudian amati hasilnya

```
itanceDemo'  
Objek dari class Pegawai dibuat  
Objek dari class Dosen dibuat  
NIP : 34329837  
Nama : Yansy Ayuningtyas  
Gaji : 3000000.0  
  
PS D:\KULIAH\SEMESTER 3\PBO\JOBSHEET\PRAKTIKUM 7> |
```

B. PERTANYAAN

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

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

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?

Jawab : Karena pada class dosen merupakan subclass dari class Pegawai yang dimana pada class Dosen juga mempunyai atribut dan method yang sama dengan class Pegawai.

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?

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


4. PERCOBAAN 3 (Hak akses)

A. TAHAPAN PERCOBAAN

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

```
public class Pegawai {  
    public String nip;  
    public String nama;  
    private double gaji;  
}
```

2. Run program kemudian amati hasilnya.
Setelah di run muncul pesan eror seperti berikut.



```
▼ J InheritanceDemo.java 1  
⚡ The field Pegawai.gaji is not visible Java(33554503) [Ln 7, Col 16]
```

3. Ubah access level modifier atribut gaji menjadi protected kemudian pindah class Pegawai kepackage baru, misalnya "testpackage".

```
package testpackage;  
  
public class Pegawai {  
    public String nip;  
    public String nama;  
    protected double gaji;  
}
```

4. Import class Pegawai dari testpackage pada class Dosen.

```
package inheritance;  
import testpackage.Pegawai;
```

5. Akses atribut gaji pada class Dosen dengan coba mencetak atribut gaji pada constructorDosen

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

6. 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?
Jawab: karena akses modifier atribut Private, yang dimana hanya bisa diakses dari class yang sama. Karena gaji memiliki akses private class Dosen yang merupakan subclass dari Pegawai tidak dapat mengakses atribut gaji secara langsung
2. Pada langkah 5, setelah class Pegawai berpindah ke package yang berbeda, class Dosen masih dapat mengakses atribut gaji. Mengapa?
Jawab: karena atribut gaji hanya bisa diakses pada class di package yang sama atau subclass di package yang berbeda
3. Berdasarkan percobaan tersebut, bagaimana menentukan atribut dan method yang akan diwariskan oleh parent class ke child class?
Jawab: menggunakan akses level modifier

6. PERCOBAAN 4 (Super - atribut)

A. TAHAPAN PERCOBAAN

1. Butlah method `getAllInfo()` pada class `Dosen`

```
public String getAllInfo() {  
    String info = "";  
    info += "NIP      : " + nip + "\n";  
    info += "Nama      : " + nama + "\n";  
    info += "Gaji      : " + gaji + "\n";  
    info += "NIDN      : " + nidn + "\n";  
  
    return info;  
}
```

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

```
public static void main(String[] args) {  
    Dosen dosen1 = new Dosen();  
  
    dosen1.nama = "Yansy Ayuningtyas";  
    dosen1.nip = "34329837";  
    dosen1.gaji = 3000000;  
    dosen1.nidn = "1989432439";  
  
    System.out.println(dosen1.getAllInfo());  
}
```

3. Run program kemudian amati hasilnya

```
aceStorage\199a1608eac4f7be9c1b1985045c1feb\redhat  
itanceDemo'  
Objek dari class Pegawai dibuat  
0.0  
Objek dari class Dosen dibuat  
NIP: 34329837  
Nama: Yansy Ayuningtyas  
Gaji: 3000000.0  
NIDN: 1989432439  
  
PS D:\KULIAH\SEMESTER 3\PBO\JOBSHEET\PRAKTIKUM 7>
```

4. Lakukan modifikasi method `getAllInfo()` pada class `Dosen`

```
public String getAllInfo() {  
    String info = "";  
    info += "NIP      : " + this.nip + "\n";  
    info += "Nama      : " + this.nama + "\n";  
    info += "Gaji      : " + this.gaji + "\n";  
    info += "NIDN      : " + this.nidn + "\n";  
  
    return info;  
}
```

5. Run program kemudian bandingkan hasilnya dengan langkah no 2.

```

1feb\redhat.java\jdt_ws\PRAKTIKUM 7_49a7851a\bin' '1r
Objek dari class Pegawai dibuat
0.0
Objek dari class Dosen dibuat
NIP: 34329837
Nama: Yansy Ayuningtyas
Gaji: 3000000.0
NIDN: 1989432439

PS D:\KULIAH\SEMESTER 3\PBO\JOBSHEET\PRAKTIKUM 7>

```

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

```

public String getAllInfo() {
    String info = "";
    info += "NIP      : " + super.nip + "\n";
    info += "Nama      : " + super.nama + "\n";
    info += "Gaji       : " + super.gaji + "\n";
    info += "NIDN      : " + super.nidn + "\n";

    return info;
}

```

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

Terjadi eror ketika program di run

```

Objek dari class Pegawai dibuat
0.0
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 Dosen.getAllInfo(Dosen.java:16)
    at InheritanceDemo.main(InheritanceDemo.java:10)
PS D:\KULIAH\SEMESTER 3\PBO\JOBSHEET\PRAKTIKUM 7>

```

8. Lakukan modifikasi method `getAllInfo()` pada class Dosen kembali

```
public String getAllInfo() {  
    String info = "";  
    info += "NIP          : " + super.nip + "\n";  
    info += "Nama          : " + super.nama + "\n";  
    info += "Gaji          : " + super.gaji + "\n";  
    info += "NIDN          : " + this.nidn + "\n";  
  
    return info;  
}
```

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

```
PS D:\KULIAH\SEMESTER 3\PBO\JOBSHEET\PRAKTIKUM 7> java -cp ..\resources\classes\InheritanceDemo.jar ..\resources\classes\InheritanceDemo  
1feb\redhat.java\jdt_ws\PRAKTIKUM 7_49a7851a\bin' 'InheritanceDemo'  
Objek dari class Pegawai dibuat  
0.0  
Objek dari class Dosen dibuat  
NIP: 34329837  
Nama: Yansy Ayuningtyas  
Gaji: 3000000.0  
NIDN: 1989432439  
  
PS D:\KULIAH\SEMESTER 3\PBO\JOBSHEET\PRAKTIKUM 7>
```

B. PERTANYAAN

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

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

2. Mengapa error terjadi pada program no 6?

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

7. PERCOBAAN 5 (super & overriding)

A. TAHAPAN PERCOBAAN

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

```
public String getAllInfo() {  
    String info = getInfo();  
    info += "NIDN          : " + nidn;  
  
    return info;  
}
```

Berikut adalah hasilnya

```
1feb\redhat.java\jdt_ws\PRAKTIKUM 7_49a7851a\bin' 'Inher  
Objek dari class Pegawai dibuat  
0.0  
Objek dari class Dosen dibuat  
NIP : 34329837  
Nama : Yansy Ayuningtyas  
Gaji : 3000000.0  
NIDN: 1989432439  
  
PS D:\KULIAH\SEMESTER 3\PBO\JOBSHEET\PRAKTIKUM 7> █
```

2. Lakukan modifikasi kembali pada method `getAllInfo()`. Run program kemudian amatihasilnya

```
public String getAllInfo() {  
    String info = this.getInfo();  
    info += "NIDN          : " + nidn;  
  
    return info;  
}
```

Berikut ini adalah hasilnya

```
Objek dari class Pegawai dibuat  
0.0  
Objek dari class Dosen dibuat  
NIP : 34329837  
Nama : Yansy Ayuningtyas  
Gaji : 3000000.0  
NIDN: 1989432439  
PS D:\KULIAH\SEMESTER 3\PBO\JOBSHEET\PRAKTIKUM 7> █
```

3. Lakukan modifikasi kembali pada method `getAllInfo()`. Run program kemudian amatihasilnya

```
public String getAllInfo() {  
    String info = super.getInfo();  
    info += "NIDN          : " + nidn;  
  
    return info;  
}
```

Berikut ini adalah hasilnya

```
1feb\redhat.java\jdt_ws\PRAKTIKUM 7_49a7851a\bin\ In  
Objek dari class Pegawai dibuat  
0.0  
Objek dari class Dosen dibuat  
NIP : 34329837  
Nama : Yansy Ayuningtyas  
Gaji : 3000000.0  
NIDN: 1989432439  
PS D:\KULIAH\SEMESTER 3\PBO\JOBSHEET\PRAKTIKUM 7> █
```

4. Tambahkan method `getInfo()` pada class `Dosen` dan modifikasi method `getAllInfo()` sebagai berikut

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

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

    public String getInfo(){
        return "NIDN      : " + this.nidn + "\n";
    }

    public String getAllInfo(){
        String info = super.getInfo();
        info += this.getInfo();

        return info;
    }
}
```

B. PERTANYAAN

1. Apakah ada perbedaan method `getInfo()` yang diakses pada langkah 1, 2, dan 3?
Jawab : tidak ada perbedaan dari langkah 1,2, dan 3
2. Apakah ada perbedaan method `super.getInfo()` dan `this.getInfo()` yang dipanggil dalam method `getAllInfo()` pada langkah 4? Jelaskan!
Jawab : Ketika `super.getInfo()` dipanggil dalam method `getAllInfo`, maka akan secara langsung memanggil implementasi `getInfo()` yang ada pada Superclass
3. Pada method manakah terjadi overriding? Jelaskan!
Jawab : Overriding terjadi ketika subclass memiliki method dengan nama dan signature yang sama dengan superclassnya. Pada class `Dosen` terdapat method `getInfo()` dengan signature yang sama dengan class `Pegawai`

```
public class Pegawai{
    public String getInfo(){
        String info = "";
        info += "NIP : " + nip + "\n";
        info += "Nama : " + nama + "\n";
        info += "Gaji : " + gaji + "\n";

        return info;
    }
}

public class Dosen extends Pegawai {
    public String getInfo(){
        return "NIDN      : " + this.nidn + "\n";
    }
}
```

4. Tambahkan keyword `final` pada method `getInfo()` di class `Pegawai`. Apakah program dapat dicompile? Mengapa?
Jawab : karena program tidak dapat dicompile karena `final` merupakan modifier yang digunakan untuk membuat class, atribut atau method tidak dapat diubah

8. PERCOBAAN 6 (overloading)

A. TAHAPAN PERCOBAAN

1. Tambahkan constructor baru untuk class `Dosen` sebagai berikut

```
public Dosen(String nip, String nama, double gaji, String nidn){
    System.out.print("Objek dari class Dosen dibuat dengan constructor berparameter");
}
```

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

```
public static void main(String[] args) {  
    Dosen dosen2 = new Dosen("34329837", "Yansy Ayuningtyas", 3000000, "1989432439");  
    System.out.println(dosen2.getAllInfo());  
}
```

B. PERTANYAAN

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

Jawab : hasil yang ditampilkan berupa nilai null

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

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

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

Jawab : pada konsep **OVERLOADING**

9. PERCOBAAN 7 (super - constructor)

A. TAHAPAN PERCOBAAN

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

```
public Dosen(String nip, String nama, double gaji, String nidn){  
    this.nip = nip;  
    this.nama = nama;  
    this.gaji = gaji;  
    this.nidn = nidn;  
}
```

Hasilnya :

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS  SEARCH  
PS D:\KULIAH\SEMESTER 3\PBO\JOBSHEET\PRAKTIKUM 8> & 'C:  
deDetailsInExceptionMessages' '-cp' 'C:\Users\me\AppData  
1a27d0169d723c0\redhat.java\jdt_ws\PRAKTIKUM 8_49a7851b'  
Objek dari class Pegawai dibuat  
NIP : 34329837  
Nama : Yansy Ayuningtyas  
Gaji : 3000000.0  
NIDN: 1989432439  
PS D:\KULIAH\SEMESTER 3\PBO\JOBSHEET\PRAKTIKUM 8>
```

2. Modifikasi constructor pada class Dosen sebagai berikut. Run program kemudian amatihasilnya.

```
public Dosen(String nip, String nama, double gaji, String nidn){  
    super.nip = nip;  
    super.nama = nama;  
    super.gaji = gaji;  
    this.nidn = nidn;  
}
```

```
PS D:\KULIAH\SEMESTER 3\PBO\JOBSHEET\PRAKTIKUM 8> &  
deDetailsInExceptionMessages' '-cp' 'C:\Users\me\AppData  
1a27d0169d723c0\redhat.java\jdt_ws\PRAKTIKUM 8_49a7851b'  
Objek dari class Pegawai dibuat  
NIP : 34329837  
Nama : Yansy Ayuningtyas  
Gaji : 3000000.0  
NIDN: 1989432439  
PS D:\KULIAH\SEMESTER 3\PBO\JOBSHEET\PRAKTIKUM 8>
```

3. Modifikasi constructor pada class Dosen sebagai berikut. Run program kemudian amatihasilnya.

```
public Dosen(String nip, String nama, double gaji, String nidn){  
    super();  
    super.nip = nip;  
    super.nama = nama;  
    super.gaji = gaji;  
    this.nidn = nidn;  
}
```

Hasilnya :

```
ata\Roaming\Code\User\workspaceStorage\bf3be35438a  
1b\bin' 'InheritanceDemo'  
Objek dari class Pegawai dibuat  
NIP : 34329837  
Nama : Yansy Ayuningtyas  
Gaji : 3000000.0  
NIDN: 1989432439  
PS D:\KULIAH\SEMESTER 3\PBO\JOBSHEET\PRAKTIKUM 8>
```

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

```
public class Pegawai {  
    public String nip;  
    public String nama;  
    public double gaji;  
  
    //    public Pegawai() {  
    //        System.out.println("Objek dari class Pegawai dibuat");  
    //    }  
  
    public Pegawai(String nip, String nama, double gaji) {  
        this.nip = nip;  
        this.nama = nama;  
        this.gaji = gaji;  
    }  
  
    public String getInfo(){  
        String info = "";  
        info += "NIP        : " + nip + "\n";  
        info += "Nama        : " + nama + "\n";  
        info += "Gaji        : " + gaji + "\n";  
  
        return info;  
    }  
}
```

```
nceDemo  
Exception in thread "main" java.lang.Error: Unresolved compilation problem:  
    Implicit super constructor Pegawai() is undefined. Must explicitly invoke another constructor  
  
    at Dosen.<init>(Dosen.java:6)  
    at InheritanceDemo.main(InheritanceDemo.java:3)  
PS D:\KULIAH\SEMESTER 3\PBO\JOBSHEET\PRAKTIKUM 8>
```


5. Modifikasi constructor pada class Dosen sebagai berikut. Run program kemudian amatihasilnya.

```
public Dosen(String nip, String nama, double gaji, String nidn){
    this.nidn = nidn;
    super(nip, nama, gaji);
}
```

```
ata\Roaming\Code\User\workspaceStorage\bf3be35438afe08df1a27d0169d723c0\redf
1b\bin' 'InheritanceDemo'
Exception in thread "main" java.lang.Error: Unresolved compilation problem:
    Constructor call must be the first statement in a constructor

    at Dosen.<init>(Dosen.java:13)
    at InheritanceDemo.main(InheritanceDemo.java:4)
PS D:\KULIAH\SEMESTER 3\PBO\JOBSHEET\PRAKTIKUM 8>
```

6. Modifikasi constructor pada class Dosen sebagai berikut. Run program kemudian amatihasilnya.

```
public Dosen(String nip, String nama, double gaji, String nidn){
    super(nip, nama, gaji);
    this.nidn = nidn;
}
```

```
ata\Roaming\Code\User\workspaceStorage\bf3be35438a
1b\bin' 'InheritanceDemo'
NIP : 34329837
Nama : Yansy Ayuningtyas
Gaji : 3000000.0
NIDN: 1989432439
PS D:\KULIAH\SEMESTER 3\PBO\JOBSHEET\PRAKTIKUM 8>
```

B. PERTANYAAN

1. Apakah terdapat perbedaan hasil pada langkah 1 dan 2? Jelaskan!
Jawab : Pada langkah 1 menggunakan **this** yang dimana merujuk pada atribut pada current class/object. Jika tidak maka baru akan merujuk ke atribut parent class. Sedangkan langkah 2 menggunakan kata kunci **super** yang merujuk langsung pada parent class/object
2. Apakah terdapat perbedaan hasil pada langkah 2 dan 3? Jelaskan!
Jawab : Perbedaannya adalah
 - **Langkah 2:** Constructor menggunakan **super.nip = nip;**, **super.nama = nama;**, dan **super.gaji = gaji;** untuk mengisi atribut dari superclass **Pegawai** secara langsung.
 - **Langkah 3:** Constructor diubah menggunakan **super();** dan tetap menggunakan **super.nip = nip;**, **super.nama = nama;**, dan **super.gaji = gaji;**, namun memanggil constructor default dari superclass dengan **super()**.
3. Mengapa terjadi error pada langkah 4?
Jawab : Error terjadi karena **constructor default (tanpa parameter)** pada superclass **Pegawai** tidak didefinisikan.
4. Apa perbedaan **super()** yang dipanggil pada langkah 3 dan 6?
Jawab : pada langkah 3 memanggil constructor default dari superclass, sedangkan pada langkah 6 yang dipanggil adalah constructor dari superclass yang memiliki parameter
5. Mengapa terjadi error pada langkah 5?
Jawab : karena kata kunci **super** dipanggil setelah **this**, yang dimana seharusnya kata kunci **super** dipanggil terlebih dahulu daripada **this**.

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.
 4. Lakukan method overriding dengan membuat method dengan nama dan signature yang sama pada parent class dan child class.
 5. Lakukan instansiasi 2 objek child class pada main class kemudian print info nya.
- Parent Class

```
1 public class BarangElektronik {
2     public String merk;
3     public String model;
4     public double harga;
5
6     public BarangElektronik() {
7         System.out.println("Ini Barang Elektronik");
8     }
9
10    public BarangElektronik(String merk, String model, double harga) {
11        this.merk = merk;
12        this.model = model;
13        this.harga = harga;
14    }
15
16    public String getInfoBarang() {
17        String info = "";
18        info += "Merk: " + merk + "\n";
19        info += "Model: " + model + "\n";
20        info += "Harga: " + harga + "\n";
21
22        return info;
23    }
24 }
```

Sub class

```
1 public class Televisi extends BarangElektronik {
2     private int ukuranLayar; // Atribut tambahan pada child class
3
4
5     public Televisi() {
6         System.out.println("Detail Televisi");
7     }
8
9
10    public Televisi(String merk, String model, double harga, int ukuranLayar) {
11        super(merk, model, harga); // Memanggil constructor berparameter dari parent
12        this.ukuranLayar = ukuranLayar;
13    }
14    public String getInfo(){
15        return "Ukuran Layar: " + this.ukuranLayar+"\n";
16    }
17    public String getAllInfo(){
18        String info = super.getInfoBarang();
19        info += "Ukuran Layar: " + this.ukuranLayar + " inc\n";
20        return info;
21    }
22 }
```

```

1  public class Kulkas extends BarangElektronik {
2      private int kapasitas;
3
4      public Kulkas() {
5          System.out.println("Detail Kulkas");
6      }
7
8      public Kulkas(String merk, String model, double harga, int kapasitas) {
9          super(merk, model, harga);
10         this.kapasitas = kapasitas;
11     }
12
13     public String getAllInfo(){
14         String info = super.getInfoBarang();
15         info += "Kapasitas isi: " + this.kapasitas + "\n";
16
17         return info;
18     }
19 }
20

```

Class Main

```

1  public class BarangElektronikDemo {
2      public static void main(String[] args) {
3          Televisi tv1 = new Televisi();
4
5          Televisi tv2 = new Televisi("Samsung", "QLED", 1000000, 55);
6          System.out.println(tv2.getAllInfo());
7
8          Kulkas kulkas1 = new Kulkas();
9          Kulkas kulkas2 = new Kulkas("LG", "GN-B202", 4000000, 210.0);
10         System.out.println(kulkas2.getAllInfo());
11     }
12 }
13
14

```

Hasil runnya

```

PROBLEMS 3 OUTPUT DEBUG CONSOLE TERMINAL PORTS SEA
PS D:\KULIAH\SEMESTER 3\PBO\JOBSHEET\PRAKTIKUM 8> & 'C:\Program Files\Java\jdk-11.0.2\bin\java.exe' -cp 'C:\Users\me\AppData\Local\Temp\1a27d0169d723c0\redhat.java\jdt_ws\PRAKTIKUM 8_49a7851b\
Ini Barang Elektronik
Detail Televisi
Merk: Samsung
Model: QLED
Harga: 1000000.0
Ukuran Layar: 55 inc

Ini Barang Elektronik
Detail Kulkas
Merk: LG
Model: GN-B202
Harga: 4000000.0
Kapasitas isi: 210

PS D:\KULIAH\SEMESTER 3\PBO\JOBSHEET\PRAKTIKUM 8>

```