M. Hasan Basri
2241760139
SIB 2C

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.

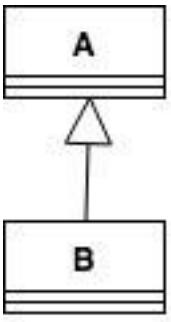
Pada bahasa pemrograman Java, deklarasi inheritance dilakukan dengan cara menambahkan kata kunci **extends** setelah deklarasi nama class, kemudian diikuti dengan nama parent class-•-nya. Kata kunci extends tersebut memberitahu kompiler Java bahwa kita ingin melakukan **extension/ perluasan** class. 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 panambahan atribut dan method khusus yang hanya dimiliki oleh class B.

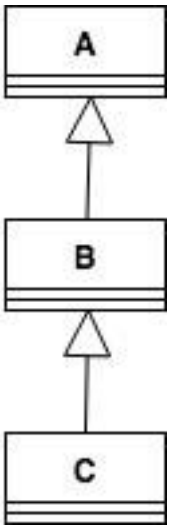
Terdapat 3 bentuk pewarisan: single inheritance, multilevel inheritance, dan multiple inheritance.

1. Single Inheritance
   Single inheritance adalah inheritance dimana suatu subclass hanya mempunyai satu parent class.



Gambar 1. Contoh Single Inheritance

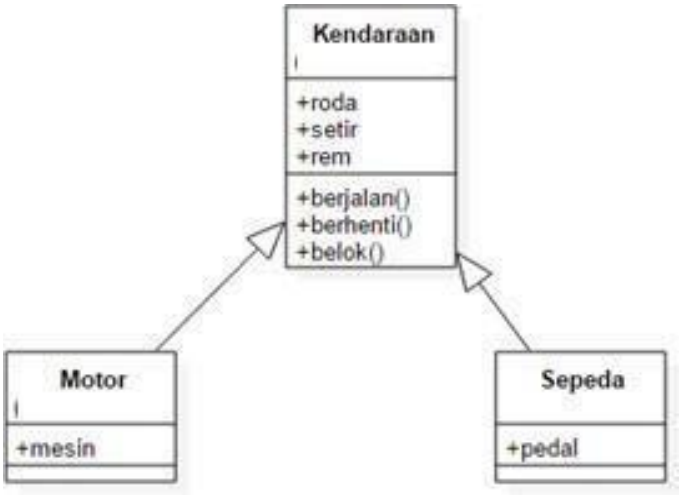
2. Multilevel Inheritance
   Multilevel inheritance adalah inheritance dengan subclass yang menjadi superclass bagi class yang lain. Contoh:



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()**
  Memganggil constructor parent, hanya dapat dilakukan pada baris pertama dalam constructor child
- **super(parameter1, parameter2,dst)**
  Memanggil constructor parent class dengan parameter, hanya dapat dilakukan pada baris pertama 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 compiletime polymorphism
   - Disebut juga static binding atau early binding
2. **Overriding**
   - Overriding terjadi ketika child class memiliki method dengan nama dan signature yang sama dengan 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:

```
Pegawai.java > Pegawai
1   public class Pegawai {
2
3       public Pegawai() {
4           System.out.println(x:"Objek dari class Pegawai dibuat");
5       }
6   }
```

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

```
Dosen.java > Dosen
1   public class Dosen extends Pegawai {
2
3       public Dosen() {
4           System.out.println(x:"Objek dari class Dosen dibuat");
5       }
6   }
```

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

```
MainP1.java > ...
1   public class MainP1 {
        Run | Debug
2       public static void main(String[] args) {
3           Dosen dosen1 = new Dosen();    The value of the local variable dosen1 is not used
4
5       }
6   }
7
```

4. Run programnya kemudian amati hasilnya.

```
Objek dari class Pegawai dibuat
Objek dari class Dosen dibuat
PS C:\KULIAH\Semester 4\OOP Praktikum\praktikum7\percobaan1(extends)>
```

B. PERTANYAAN

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

   Jawab : Dalam percobaan di atas, class Dosen adalah child class (subclass) sedangkan class Pegawai adalah parent class (superclass).

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

   Jawab : Kata kunci yang membuat child class dan parent class tersebut memiliki relasi adalah kata kunci extends. Dalam Java, extends digunakan untuk menunjukkan bahwa sebuah class merupakan turunan dari class lainnya, sehingga class yang menggunakan extends akan menjadi subclass dari class yang di-extends.

3. Berdasarkan hasil yang ditampilkan oleh program, ada berapa constructor yang dieksekusi? Constructor class mana yang lebih dulu dieksekusi?
   Jawab : Terdapat dua constructor yang dieksekusi. Constructor dari class Pegawai dieksekusi lebih dulu, kemudian constructor dari class Dosen dieksekusi. Hal ini terjadi karena ketika objek dari class Dosen diciptakan, Java secara otomatis memanggil constructor dari superclassnya (Pegawai) sebelum menjalankan constructor dari class Dosen itu sendiri.

A. Percobaan 2(Pewarisan)

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

Pegawai.java
```java
public class Pegawai {
    public String nip;
    public String nama;
    public double gaji;

    public Pegawai() {
        System.out.println(x:"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

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

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

3.Pada class InheritanceDemo.java tuliskan kode berikut

```java
public class InheritanceDemo {
    Run | Debug
    public static void main(String[] args) {
        Dosen dosen1 = new Dosen();

        dosen1.nama = "Yansy Ayuningtyas";
        dosen1.nip = "34329837";
        dosen1.gaji = 3000000 ;
        dosen1.nidn = "1989432432439";

        System.out.println(dosen1.getInfo());

    }
}
```

4.Run program kemudian amati hasilnya

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

PS C:\KULIAH\Semester 4\OOP Praktikum\praktikum7\KodePercobaan>
```

B. PERTANYAAN

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

   Jawab : Program berhsil dijalankan dengan baik.

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 dalam konsep pewarisan (inheritance), subclass (Dosen) akan mewarisi atribut-atribut dan method-method dari superclass (Pegawai). Karena atribut nip, nama, dan gaji beserta method getInfo() sudah ada dan diwarisi oleh class Dosen, maka objek dosen1 dapat mengakses serta mengisi nilai atribut tersebut.

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 meskipun class Dosen tidak memiliki deklarasi langsung untuk method getInfo(), namun class Dosen mewarisi method tersebut dari superclassnya, yaitu class Pegawai. Jadi, objek dosen1 dapat memanggil method getInfo() karena method tersebut sudah ada dalam hierarki kelas yang diwarisi oleh class Dosen.

A. Percobaan 3 (Hak akses)

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.

```
The field Pegawai.gaji is not visible
```

3. Ubah access level modifier atribut gaji menjadi protected kemudian pindah class Pegawai ke package baru, misalnya “testpackage”.

```
package testPackage;

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

4. import class pegawai dari testpackage pada dosen.

```
Inheritance > J Dosen.java > ...
1   package Inheritance;
2   import testPackage.Pegawai;
```

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

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

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

6. Ubah kembali access level modifier menjadi public dan kembalikan class Pegawai ke package semula.

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

PS C:\KULIAH\Semester 4\OOP Praktikum\praktikum7\KodePercobaan>
```

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 : terjadi error karena ketika atribut gaji diubah access level modifiernya menjadi private, maka atribut tersebut hanya dapat diakses secara langsung oleh class Pegawai sendiri. Tidak dapat diakses oleh class lain, termasuk subclass seperti Dosen.

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

   Jawab : setelah class Pegawai dipindahkan ke package yang berbeda, class Dosen masih dapat mengakses atribut gaji karena access level modifier yang digunakan adalah protected. Dengan access level modifier protected, atribut dapat diakses oleh class yang berada dalam package yang sama maupun subclass yang berada dalam package yang berbeda.

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

   Jaweb : Untuk menentukan atribut dan method yang akan diwariskan oleh parent class ke child class, kita perlu memperhatikan access level modifier-nya. Jika ingin sebuah atribut atau method dapat diakses oleh child class, maka access level modifier-nya harus memungkinkan untuk diakses oleh child class, seperti public atau protected. Jika access level modifier-nya adalah private, maka atribut atau method tersebut hanya dapat diakses oleh class yang mendeklarasikannya, dan tidak akan diwariskan ke child class. Dalam hal ini, perlu dipertimbangkan secara hati-hati dalam mendesain struktur kelas agar pewarisan (inheritance) dapat berjalan dengan baik sesuai kebutuhan.

A. (Super - atribut)

1. Buatlah 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 dosen 1 pada class InheritanceDemo.java

```
public class InheritanceDemo {
    Run | Debug
    public static void main(String[] args) {
        Dosen dosen1 = new Dosen();

        dosen1.nama = "Yansy Ayuningtyas";
        dosen1.nip = "34329837";
        dosen1.gaji = 3000000 ;
        dosen1.nidn = "1989432432439";

        System.out.println(dosen1.getAllInfo());
```

3. Run program kemudian amati hasilnya

```
Objek dari class Pegawai dibuat
Objek dari class Dosen dibuat
NIP    :  34329837
Nama   :  Yansy Ayuningtyas
Gaji   : 3000000.0
NIDN   : 1989432432439

PS C:\KULIAH\Semester 4\OOP Praktikum\praktikum7\KodePercobaan>
```

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.

```
Objek dari class Pegawai dibuat
Objek dari class Dosen dibuat
NIP    :  34329837
Nama   :  Yansy Ayuningtyas
Gaji   : 3000000.0
NIDN   : 1989432432439

PS C:\KULIAH\Semester 4\OOP Praktikum\praktikum7\KodePercobaan>
```

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;
}
```

Error

```
info += "NIDN   : " + super.nidn + "\n";    nidn cannot be resolved or is not a field
```

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

```
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 Inheritance.Dosen.getAllInfo(Dosen.java:14)
        at Inheritance.InheritanceDemo.main(InheritanceDemo.java:11)
PS C:\KULIAH\Semester 4\OOP Praktikum\praktikum7\KodePercobaan>
```

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 progam pada no 2 dan no 4.

```
Objek dari class Pegawai dibuat
Objek dari class Dosen dibuat
NIP    :  34329837
Nama   :  Yansy Ayuningtyas
Gaji   : 3000000.0
NIDN   : 1989432432439

PS C:\KULIAH\Semester 4\OOP Praktikum\praktikum7\KodePercobaan>
```

B. PERTANYAAN

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

   Jawab : terdapat perbedaan hasil nama, nip, dan gaji yang ditampilkan pada program 1, 4, dan 8. Pada program 1, atribut nip, nama, dan gaji diambil langsung dari instance Dosen karena tidak disertakan super. sebelum pemanggilan atribut. Pada program 4, karena menggunakan this. sebelum pemanggilan atribut nidn, maka akan terjadi error karena nidn tidak didefinisikan di kelas Dosen, tetapi di kelas Pegawai. Pada program 8, super. digunakan sebelum pemanggilan atribut nip, nama, dan gaji yang mengambil nilai dari superclass, sedangkan this. digunakan sebelum pemanggilan atribut nidn yang merupakan atribut langsung dari kelas Dosen.

2. Mengapa error terjadi pada program no 6?

   Jawab : Error terjadi karena terdapat kesalahan sintaksis pada bagian pemanggilan atribut nidn. Penggunaan super. menunjukkan bahwa Anda ingin mengakses atribut dari superclass, tetapi nidn adalah atribut dari kelas Dosen bukan dari superclass-nya. Sehingga, tidak dapat menggunakan super. untuk mengakses atribut tersebut.

PERCOBAAN 5 (super & overriding)

A. TAHAPAN PERCOBAAN

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

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

    return info;

}
```

Output:

```
Objek dari class Pegawai dibuat
Objek dari class Dosen dibuat
NIP: 34329837
Nama: Yansy Ayuningtyas
Gaji: 3000000.0
NIDN   : 1989432432439
PS C:\KULIAH\Semester 4\OOP Praktikum\praktikum7\KodePercobaan>
```

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

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

    return info;

}
```

Output :

```
Objek dari class Pegawai dibuat
Objek dari class Dosen dibuat
NIP: 34329837
Nama: Yansy Ayuningtyas
Gaji: 3000000.0
NIDN   : 1989432432439
PS C:\KULIAH\Semester 4\OOP Praktikum\praktikum7\KodePercobaan>
```

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

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

    return info;

}
```

Output :

```
Objek dari class Pegawai dibuat
Objek dari class Dosen dibuat
NIP: 34329837
Nama: Yansy Ayuningtyas
Gaji: 3000000.0
NIDN   : 1989432432439
PS C:\KULIAH\Semester 4\OOP Praktikum\praktikum7\KodePercobaan>
```

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

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

    public Dosen() {
        System.out.println(x:"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 : Pada langkah 1, 2, dan 3, yang diakses adalah method getInfo() dari superclass Pegawai, namun cara pemanggilannya berbeda-beda. Pada langkah 1, method tersebut dipanggil tanpa menggunakan this atau super. Pada langkah 2, method tersebut dipanggil menggunakan this, sedangkan pada langkah 3, method tersebut dipanggil menggunakan super.

2. Apakah ada perbedaan method super.getInfo() dan this.getInfo() yang dipanggil dalam method getAllInfo() pada langkah 4? Jelaskan!
   Jawab : Pada langkah 4, perbedaan antara super.getInfo() dan this.getInfo() terletak pada objek yang memanggil method tersebut. super.getInfo() memanggil method getInfo() dari superclass (Pegawai), sedangkan this.getInfo() memanggil method getInfo() dari objek saat ini (Dosen).

3. Pada method manakah terjadi overriding? Jelaskan!
   Jawab: Overriding terjadi pada method getInfo() di class Dosen. Ini terjadi ketika method dengan nama yang sama seperti di superclass diimplementasikan di subclass dengan implementasi yang berbeda. Dalam hal ini, method getInfo() di class Dosen menggantikan (override) method getInfo() di class Pegawai.

4. Tambahkan keyword final pada method getInfo() di class Pegawai. Apakah program dapat dicompile? Mengapa?

   ```java
   public final String getInfo() {
   ```
   >> Class pegawai

   ```java
   public String getInfo(){    Cannot override the final method from Pegawai
   ```
   >>class Dosen

   Jawab : Jika ditambahkan keyword final pada method getInfo() di class Pegawai, program tidak akan dapat dikompilasi. Ini disebabkan karena keyword final mengindikasikan bahwa method tersebut tidak dapat di-override oleh subclass. Namun, pada langkah 3 kita telah melakukan overriding pada method getInfo() di class Dosen, sehingga penambahan keyword final akan menghasilkan konflik. Karena itu, program tidak akan bisa dikompilasi.

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());
}
```

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

PS C:\KULIAH\Semester 4\OOP Praktikum\praktikum7\KodePercobaan>
```

B. PERTANYAAN

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

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

PS C:\KULIAH\Semester 4\OOP Praktikum\praktikum7\KodePercobaan>
```

Jawab : Hal ini terjadi karena dalam constructor baru yang telah distambahkan pada class Dosen, Anda hanya mencetak pesan "Objek dari class Dosen dibuat", tetapi tidak ada proses penginisialisasian nilai atribut nip, nama, gaji, dan nidn dengan nilai-nilai yang diterima sebagai argumen.

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

Jawab : Tidak, constructor tanpa parameter dan constructor class Dosen yang dibuat pada langkah 1 tidak memiliki signature yang sama. Constructor tanpa parameter tidak memiliki parameter sedangkan constructor yang dibuat pada langkah 1 memiliki empat parameter, yaitu nip, nama, gaji, dan nidn.

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 : Konsep yang membolehkan suatu class memiliki constructor atau method dengan nama yang sama tetapi dengan signature yang berbeda disebut dengan overloading. Overloading memungkinkan kita untuk memiliki beberapa method atau constructor dengan nama yang sama di dalam satu class, namun dengan parameter yang berbeda, sehingga kita dapat melakukan berbagai tugas berdasarkan jenis dan jumlah parameter yang berbeda.

PERCOBAAN 7 (super - constructor)

A. TAHAPAN PERCOBAAN

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

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

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

PS C:\KULIAH\Semester 4\OOP Praktikum\praktikum7\KodePercobaan\Inheritance>
```

2. Modifikasi Constructor pada class dosen sebagai berikut. Run program kemudian amati hasilnya.

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

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

PS C:\KULIAH\Semester 4\OOP Praktikum\praktikum7\KodePercobaan\Inheritance>
```

3. Modifikasi Constructor pada class dosen sebagai berikut. Run program kemudian amati hasilnya.

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

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

PS C:\KULIAH\Semester 4\OOP Praktikum\praktikum7\KodePercobaan\Inheritance>
```

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;
    }
}
```

```
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:4)
        at InheritanceDemo.main(InheritanceDemo.java:13)
PS C:\KULIAH\Semester 4\OOP Praktikum\praktikum7\KodePercobaan>
```

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

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

```
this.nidn = nidn;
super(nip, nama, gaji);    Constructor call must be the first statement in a constructor
```

```
Exception in thread "main" java.lang.Error: Unresolved compilation problems:
        Implicit super constructor Pegawai() is undefined. Must explicitly invoke another constructor
        Constructor call must be the first statement in a constructor

        at Dosen.<init>(Dosen.java:4)
        at InheritanceDemo.main(InheritanceDemo.java:13)
PS C:\KULIAH\Semester 4\OOP Praktikum\praktikum7\KodePercobaan>
```

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

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

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

PS C:\KULIAH\Semester 4\OOP Praktikum\praktikum7\KodePercobaan>
```

PERTANYAAN

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

   Jawab : Perbedaan antara langkah 1 dan 2 adalah cara penginisialisasian atribut-atribut dalam constructor. Pada langkah 1, atribut-atribut diinisialisasi langsung menggunakan this, sedangkan pada langkah 2, atribut-atribut diinisialisasi menggunakan super dengan menambahkan prefix super..

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

   Jawab : Perbedaan antara langkah 2 dan 3 adalah penambahan pemanggilan super() di dalam constructor pada langkah 3. Pemanggilan super() ini akan secara eksplisit memanggil constructor dari superclass Pegawai yang tidak memiliki parameter. Sehingga, constructor tanpa parameter dari superclass Pegawai akan dijalankan ketika constructor class Dosen dijalankan.

3. Mengapa terjadi error pada langkah 4?

   Jawab : Error pada langkah 4 terjadi karena setelah menghapus constructor tanpa parameter dari class Pegawai, tidak ada constructor default yang tersedia lagi. Ketika Anda membuat objek Dosen, maka Java akan mencoba memanggil constructor default dari superclass Pegawai (karena tidak ada constructor yang ditentukan), tetapi karena constructor default tersebut sudah dihapus, maka terjadi error.

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

   Jawab : Pada langkah 3, super() digunakan untuk memanggil constructor tanpa parameter dari superclass Pegawai, sedangkan pada langkah 6, super(nip, nama, gaji) digunakan untuk memanggil constructor superclass Pegawai yang memiliki parameter nip, nama, dan gaji. Dengan demikian, perbedaannya terletak pada constructor mana yang dipanggil dari superclass Pegawai.

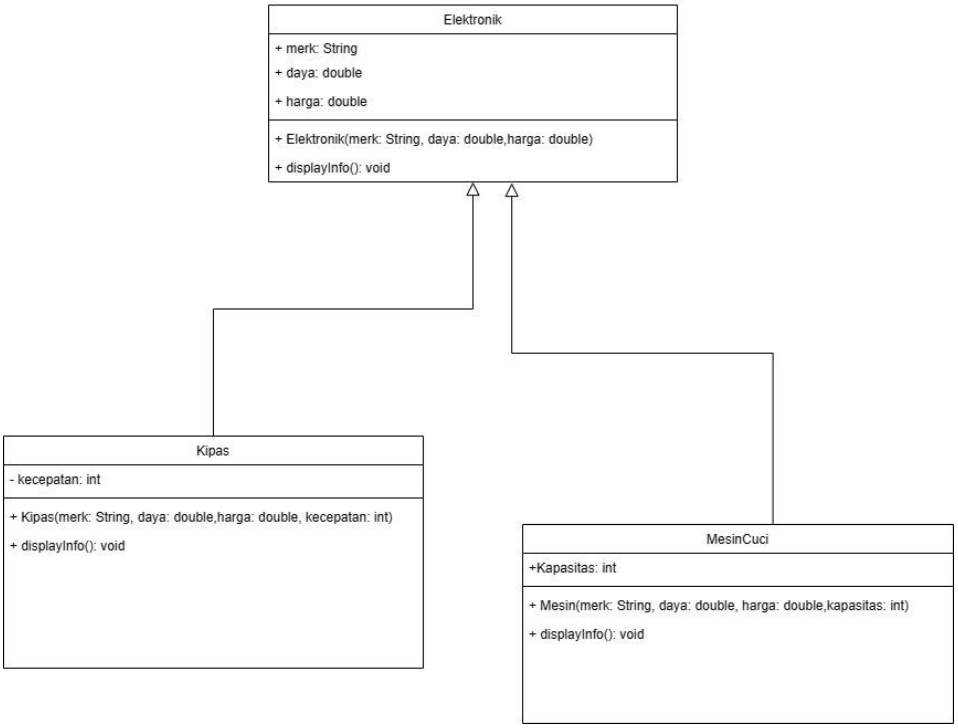
5. Mengapa terjadi error pada langkah 5?
   Jawab : Error pada langkah 5 terjadi karena pemanggilan super(nip, nama, gaji) harus dilakukan sebagai statement pertama dalam constructor class Dosen. Namun, dalam langkah tersebut, pemanggilan super(nip, nama, gaji) tidak berada di posisi pertama dalam constructor, sehingga terjadi error.

4. 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.

Nama : M. Hasan Basri
Kelas : SIB 2C
NIM : 2241760139

Macam Elektronic



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.

```
// Overloading constructor tanpa parameter
public Kipas() {
    super();
}

// Constructor berparameter
public Kipas(String merk, double daya, double harga, int kecepatan) {
    super(merk, daya, harga); // Memanggil constructor berparameter dari kelas induk
    this.kecepatan = kecepatan;
}
```

4. Lakukan method overriding dengan membuat method dengan nama dan signature yang sama pada parent class dan child class.

```java
    @Override
    public void displayInfo() {
        super.displayInfo();
        System.out.println("Kecepatan: " + kecepatan + " RPM");
    }
}
```

5. Lakukan instansiasi objek child class pada main class kemudian print info nya.

```
Informasi Kipas:
Merk: Kipas angin polytron
Daya: 50.0 Watt
Harga: Rp 250000.0
Kecepatan: 1200 RPM

Informasi Mesin Cuci:
Merk: Mesin Cuci Sharp
Daya: 500.0 Watt
Harga: Rp 1500000.0
Kapasitas: 7 kg
PS C:\KULIAH\Semester 4\OOP Praktikum\praktikum7\KodePercobaan\Inheritance>
```