

# **TUGAS 2 PEMROGRAMAN MOBILE**

**Inheritance, Interface, Komposisi, Overloading dan Overriding**

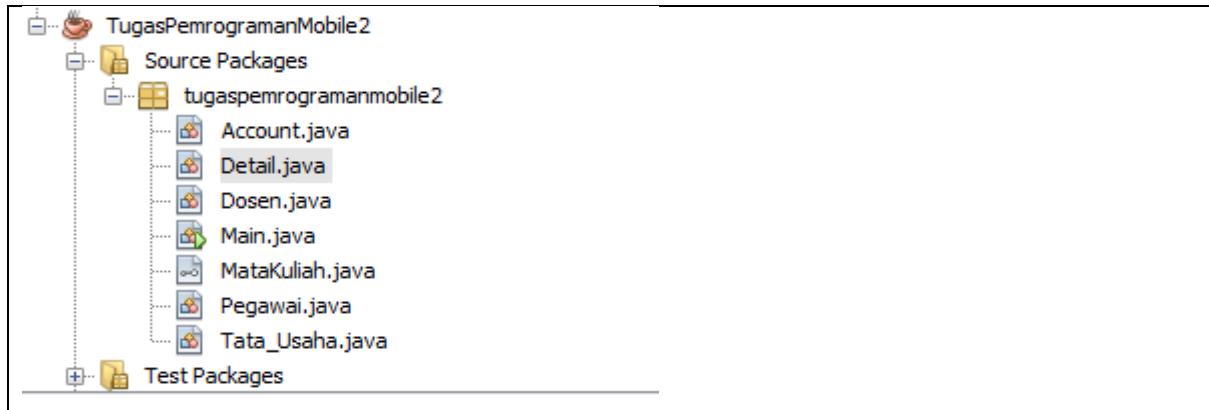


Nama : I Nyoman Krisna Pranata  
NIM : 1605551114  
Mata Kuliah : Pemrograman Mobile (D)

**TEKNOLOGI INFORMASI  
FAKULTAS TEKNIK  
UNIVERSITAS UDAYANA**

## Inheritance

Inheritance merupakan pewarisan atribut dan method pada sebuah class yang diperoleh dari class yang telah terdefiniskan tersebut. Class yang diwarisi class yang lain di sebut dengan subclass, sedangkan class mewarisi disebut dengan superclass.



Gambar diatas menunjukkan ada 7 kelas yang berbeda yaitu class Account, Detail, Dosen, Main, MataKuliah, Pegawai, dan Tata\_Usaha.

Pada pembahasan kali ini yaitu inheritance, saya akan menunjukkan 3 kelas saja sebagai contoh yaitu Pegawai sebagai parent class, dan sebagai sub class Dosen, dan Class Main untuk menjalankan programnya, berikut merupakan contoh kode programnya.

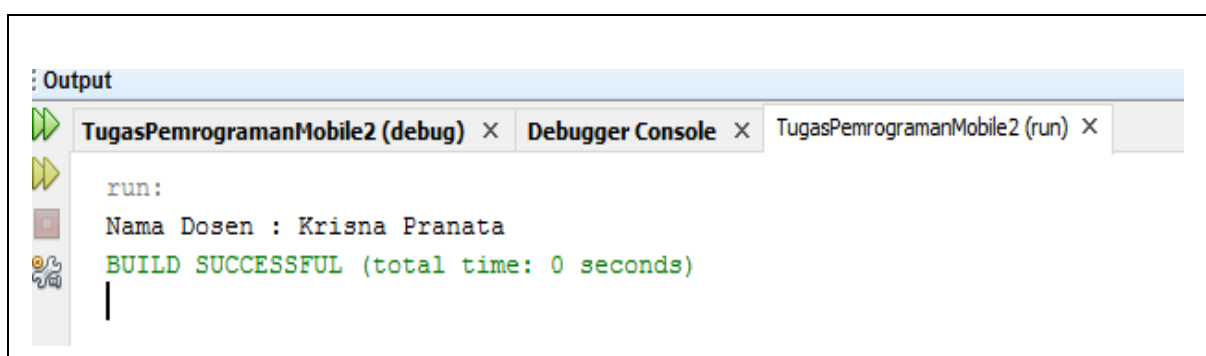
<pre>1 package tugaspemrogramanmobile2; 2 3 4 public class Pegawai { 5     private String nama; 6 7     public Pegawai() { 8     } 9     public Pegawai (String nama){ 10         this.nama = nama; 11     } 12     public String getNama() { 13         return nama; 14     } 15 16     public void setNama(String nama) { 17         this.nama = nama; 18     } 19 20     public int getGajiPokok() { 21         return 2000000; 22     } 23 }</pre>	<pre>1 package tugaspemrogramanmobile2; 2 3 4 public class Dosen extends Pegawai{ 5     private int jumlahSKS; 6     private Account account; 7     private String addAccount; 8     private final int hargaPerSKS = 100000; 9 10     public Dosen() { 11     } 12     public Dosen(String nama){ 13         super(nama); 14     } 15     public void setJumlahSKS(int SKS) { 16         this.jumlahSKS = SKS; 17     } 18     public void addAccount(Account account){ 19         this.account = account; 20     } 21     public String getAddAccount(){ 22         return account.getNama(); 23     } 24     @Override 25     public int getGajiPokok() { 26         int total_gaji = jumlahSKS*hargaPerSKS; 27         return total_gaji+super.getGajiPokok(); 28     } 29 }</pre>
--	---

Kedua Gambar diatas merupakan class Pegawai dan class Dosen, pada gambar sebelah kiri class Pegawai mendeklarasikan method setNama, getNama, dan getGajiPokok, pada

gambar diatas, dan pada gambar sebelah kanan terlihat class Dosen menggunakan extends pada parent class yaitu class pegawai, dan pada line 12 terlihat class Dosen memakai method dari class Pegawai yaitu setName dan getName, jadi tanpa harus dideklarasikan lagi setName dan getName sudah bisa dipakai oleh class Dosen.

```
1 package tugasprogramanmobile2;
2
3 public class Main {
4     public static void main(String[] args) {
5         // Detail det = new Detail();
6         // det.setName("Krisna Pranata");
7         // det.addAccount(new Account("Khryсна"));
8         // det.setMatakuliah("Pemrograman Mobile", "P3Z2");
9         // det.setJumlahSKS(12);
10        // det.print();
11
12        Dosen dosen = new Dosen();
13        dosen.setName("Krisna Pranata");
14        System.out.println(" Nama Dosen : " + dosen.getName());
15    }
16 }
17
```

Gambar diatas merupakan class main, pada gambar diatas terlihat pada line 12 mendeklarasikan class Dosen dengan variable “dosen”, dan pada line 13 terlihat dosen memakai method setName yang awalnya method tersebut terdapat pada parent class, dan pada line 14 menampilkan nama dosen dengan method yang getName yang awalnya dari parent class juga.



```
Output
TugasProgramanMobile2 (debug) x Debugger Console x TugasProgramanMobile2 (run) x
run:
Nama Dosen : Krisna Pranata
BUILD SUCCESSFUL (total time: 0 seconds)
```

Gambar diatas merupakan hasil running dari ketiga kelas tersebut, dan menampilkan nama dosen “Krisna Pranata”.

## Interface

Di dalam bahasa Java, sebuah subclass merupakan turunan langsung dari satu superclass (single inheritance). Java tidak memperbolehkan sebuah subclass diturunkan

langsung dari beberapa macam superclass (multiple inheritance). Dengan menggunakan interfes (interface), akan diperoleh efek dari multiple inheritance.

Class modifier implements digunakan untuk mendeklarasikan sebuah kelas (class) yang akan mengimplementasikan satu atau lebih interface Java. Apa bila interface Java yang akan diimplementasikan lebih dari satu maka pendeklarasiannya dipisahkan dengan tanda koma. Interface adalah kumpulan method yang hanya memuat deklarasi dan struktur method, tanpa detail implementasinya. Sedangkan detail dari method berada pada class yang mengimplementasikan interface tersebut. Interface digunakan jika ingin mengaplikasikan suatu method yang spesifik, yang tidak diperoleh dari proses inheritance. Interface tidak berisi perintah kepada system untuk melakukan sesuatu, interface hanya memetakan apa yang harus dikirimkan dan apa yang diharapkan untuk dikembalikan. Tipe data yang boleh pada interface hanya tipe data konstan. Berikut merupakan ontoh deklarasi dari interface.

Untuk lebih mengenal apa itu interface, disini saya akan menampilkan contoh kode program dari interface disini saya akan menampilkan 4 kelas yang berbeda yaitu class Dosen, Detail, Matakuliah dan class Main.

<pre>1 package tugasprogramanmobile2; 2 3 4 public class Dosen extends Pegawai{ 5     private int jumlahSKS; 6     private Account account; 7     private String addAccount; 8     private final int hargaPerSKS = 100000; 9 10    public Dosen(){ 11    } 12    public Dosen(String nama){ 13        super(nama); 14    } 15    public void setJumlahSKS(int SKS){ 16        this.jumlahSKS = SKS; 17    } 18    public void addAccount(Account account){ 19        this.account = account; 20    } 21    public String getAddAccount(){ 22        return account.getNama(); 23    } 24    @Override 25    public int getGajiPokok(){ 26        int total_gaji = jumlahSKS*hargaPerSKS; 27        return total_gaji+super.getGajiPokok(); 28    } 29 }</pre>	<pre>1 package tugasprogramanmobile2; 2 3 4 public class Detail extends Dosen implements Matakuliah{ 5     private String namaMatakuliah; 6     private String kodeMatakuliah; 7     public Detail(){ 8     } 9 10    public Detail(String nama){ 11        super(nama); 12    } 13 14    @Override 15    public void setMatakuliah(String namaMatakuliah, String kodeMatakuliah){ 16        this.namaMatakuliah = namaMatakuliah; 17        this.kodeMatakuliah = kodeMatakuliah; 18    } 19    public void print(){ 20        System.out.println("Nama Dosen : " + super.getNama()); 21        System.out.println("Nama Akun : " + super.getAddAccount()); 22        System.out.println("Nama Matakuliah : " + namaMatakuliah); 23        System.out.println("Kode Matakuliah : " + kodeMatakuliah); 24        System.out.println("Gaji Dosen : " + super.getGajiPokok()); 25    } 26 } 27 28 29 }</pre>
---	--

Gambar diatas merupakan class Dosen dan Class Detail, disini kelas Dosen sebagai parent class dari class Detail, dan gambar disebelah kanan merupakan class Detail, pada class Detail disini terdapat method setMatakuliah, fungsi ini merupakan fungsi implementasi dari kelas Matakuliah, seperti pada gambar dibawah ini.

```
1 package tugasprogramanmobile2;
2
3 public interface Matakuliah {
4     public void setMatakuliah(String namaMatakuliah, String kodeMatakuliah);
5 }
6
```

Gambar diatas merupakan interface dari Matakuliah, jadi pada interface matakuliah ini saya membuat fungsi bernama setMatakuliah yang didalamnya ada 2 parameter yaitu nama matakuliah dan kode matakuliah, yang nantinya fungsi ini dibawa ke dalam class Detail,

```
1 package tugasprogramanmobile2;
2
3 public class Main {
4     public static void main(String[] args) {
5         Detail det = new Detail();
6         det.setNama("Krisna Pranata");
7         // det.addAccount(new Account("Khrysna"));
8         det.setMatakuliah("Pemrograman Mobile", "P3Z2");
9         // det.setJumlahSKS(12);
10        det.print();
11    }
12 }
13
14
```

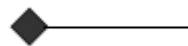
Gambar diatas merupakan gambar dari class Main, dimana pada class ini memanggil class Detail, dan memanggil method setNama, setMatakuliah, dan print didalamnya.

```
TugasProgramanMobile2 (debug) x Debugger Console x TugasProgramanMobile2 (run) x
run:
Nama Dosen : Krisna Pranata
Nama Matakuliah : Pemrograman Mobile
Kode Matakuliah : P3Z2
BUILD SUCCESSFUL (total time: 0 seconds)
```

Gambar diatas merupakan hasil running dari class Main, disana terdapat nama dosen, nama matakuliah, dan kode matakuliah.

## Komposisi

Jika sebuah class tidak bisa berdiri sendiri dan harus merupakan bagian dari class yang lain, maka class tersebut memiliki relasi Komposisi terhadap class tempat dia bergantung tersebut. Sebuah relationship composition digambarkan sebagai garis dengan ujung berbentuk jajaran genjang berisi/solid.



Komposisi merupakan hubungan yang paling kuat antar kelas. Komposisi digunakan untuk mengambil seluruh bagian hubungan. Hanya boleh ada satu komposisi dalam satu waktu. Antara instance suatu kelas dengan instance lainnya selalu terkoneksi (linked)

<pre>1 package tugasprogramanmobile2; 2 3 public class Dosen extends Pegawai{ 4     private int jumlahSKS; 5     private Account account; 6     private String addAccount; 7     private final int hargaPerSKS = 100000; 8 9     public Dosen() { 10 11     } 12     public Dosen(String nama){ 13         super(nama); 14     } 15     public void setJumlahSKS(int SKS){ 16         this.jumlahSKS = SKS; 17     } 18     public void addAccount(Account account){ 19         this.account = account; 20     } 21     public String getAddAccount(){ 22         return account.getNama(); 23     } 24     @Override 25     public int getGajiPokok(){ 26         int total_gaji = jumlahSKS*hargaPerSKS; 27         return total_gaji+super.getGajiPokok(); 28     } 29 }</pre>	<pre>1 package tugasprogramanmobile2; 2 3 public class Account{ 4     private String nama; 5 6     public String getNama() { 7         return nama; 8     } 9 10    public void setNama(String nama) { 11        this.nama = nama; 12    } 13    public Account(){ 14 15    } 16    public Account(String nama){ 17        this.nama = nama; 18    } 19 } 20 }</pre>
---	--

Gambar diatas menunjukan 2 kelas yaitu kelas Dosen dan kelas Account, terlihat pada kelas Dosen, memanggil kelas Account dengan method addAccount dan terdapat method getAddAccount, yang nantinya jika kelas Account tidak dipanggil maka hasilnya akan null, tetapi jika kelas Dosen tidak dipanggil maka kelas Account tidak akan terpanggil juga.

```

1  package tugasprogramanmobile2;
2
3  public class Main {
4      public static void main(String[] args) {
5          Detail det = new Detail();
6          // det.setNama("Krisna Pranata");
7          // det.addAccount(new Account("Khrysna"));
8          // det.setMatakuliah("Pemrograman Mobile", "P3Z2");
9          // det.setJumlahSKS(12);
10         // det.print();
11
12         Dosen dosen = new Dosen();
13         dosen.setNama("Krisna Pranata");
14         dosen.addAccount(new Account("Khrysna"));
15         System.out.println("Nama Dosen : " + dosen.getNama());
16         System.out.println("Nama Akun : " + dosen.getAddAccount());
17     }
18 }
19

```

Gambar diatas menunjukkan class Main, dimana kelas dosen terpanggil dan terdapat method setNama, disini terdapat method addAccount dimana pada method ini harus memanggil kelas Account terlebih dahulu, lalu baru bisa mengisi parameternya, jadi pada dasarnya jika kita tidak memanggil kelas dosen maka kelas Account pun nanti tidak akan terbaca pada program, seperti itulah komposisi hubungannya sangat erat kepada kelas parent nya.

## Overloading

Overloading adalah suatu proses menggunakan nama yang sama untuk dua atau lebih fungsi. Setiap definisi ulang dari fungsi yang di overloading harus menggunakan tipe parameter, urutan parameter, atau jumlah parameter yang berbeda. Jumlah, tipe atau urutan parameter dari suatu fungsi disebut function signature. Jika kita memiliki sejumlah fungsi dengan nama yang sama, compiler akan mengidentifikasi fungsi-fungsi tersebut berdasarkan parameternya. Method overloading digunakan untuk membuat beberapa fungsi (method) dengan nama yang sama dan mengerjakan tugas yang mirip.

Tujuannya, agar programmer tidak kesulitan dalam mengingat sebuah fungsi yang tugasnya mirip. Misalnya untuk membuat fungsi perkalian. Ada perkalian yang membutuhkan dua argumen dan ada perkalian yang membutuhkan tiga argumen (tugasnya mirip, yaitu sama-sama mengalikan argumen, hanya jumlah argumen yang berbeda). Kalau dibuat fungsi dengan nama yang berbeda, kemungkinan programmer akan repot. Jadi, bagaimana cara compiler membedakan method tersebut, padahal namanya sama? Compiler akan memilih fungsi dengan

mengamati jumlah, tipe data dan urutan argumen. Nah, dari sini jelas bahwa compiler menggunakan daftar parameter untuk membedakan fungsi dengan nama yang sama. Bukan membedakan dengan return type-nya (nilai balik).

Sehingga, untuk membuat method overloading diperlukan setidaknya satu dari tiga syarat di bawah ini:

1. mempunyai jumlah parameter berbeda.
2. mempunyai tipe data dari parameter yang berbeda.
3. mempunyai urutan tipe data yang berbeda.

Berikut saya buat contoh program sederhana dengan menggunakan metode overloading:

```
2
3 public class OverLoading {
4     public void times(int a, int b){ // membuat fungsi perkalian dengan mengalikan 2 tipe data
5         System.out.println("Hasil perkaliannya adalah "+(a*b));
6     }
7     public void times(int a, int b, int c){ // membuat fungsi perkalian dengan mengalikan 3 tipe data
8         System.out.println("Hasil perkaliannya adalah "+(a*b*c));
9     }
10    public static void main(String[] args) {
11        OverLoading ol = new OverLoading();//memanggil kelas OverLoading dan memberni nama objek yaitu ol
12        ol.times(4, 8); //memanggil fungsi times dengan 2 tipe data
13        ol.times(3, 4, 6); //memanggil fungsi times dari kelas OverLoading dengan 3 tipe data
14    }
15 }
16
```

Output:



```
Output X
JavaApplication6 (run) #2 X JavaApplication6 (run) #3 X JavaApplication6 (run) #4 X
run:
Hasil perkaliannya adalah 32
Hasil perkaliannya adalah 120
BUILD SUCCESSFUL (total time: 0 seconds)
```

Gambar diatas menunjukan sebuah program untuk mengalikan sebuah bilangan, disini saya membuat 2 buah fungsi atau method dengan nama fungsi yang sama yaitu times, tetapi kedua fungsi times ini memiliki parameter yang berbeda di tiap fungsinya, pada fungsi times pertama terlihat fungsi times ini hanya mengalikan 2 bilangan saja, dan fungsi times yang kedua mengalikan 3 bilangan setelah itu pada public main saya memanggil kelas OverLoading dengan nama objek ol, dan pada line ke 12 saya memasukkan 2 angka, yang nantinya dibaca



oleh system dan memilih fungsi times dengan 2 parameter juga, setelah itu saya memanggil di line ke 13 sama seperti line 12, yang nantinya dia memanggil fungsi times yang kedua dimana terdapat 3 parameter.

## Overriding

Overriding adalah method subclass sama dengan method super class, parameter yang dimiliki sama tetapi pernyataan atau implementasinya berbeda. Atau overriding dapat diartikan merupakan suatu keadaan dimana kelas anak dapat mengubah atau bisa kita bilang memodifikasi atau memperluas data dan method pada kelas induk. Keuntungan Overriding adalah dapat menambahkan sifat / atribut pada kelas induk nya lebih Jelasnya Overloading secara singkat :

1. Mendefinisikan method dengan nama sama pada class yang berbeda.
2. Konsep dapat di manfaatkan dalam beberapa kasus inheritance, di mana menginginkan penamaan method dengan nama sama namun berbeda dalam implemenasinya.

Untuk lebih jelasnya saya akan menampilkan 3 kelas yang berbeda yaitu kelas Pegawai, Dosen, dan Main, seperti pada gambar dibawah ini.

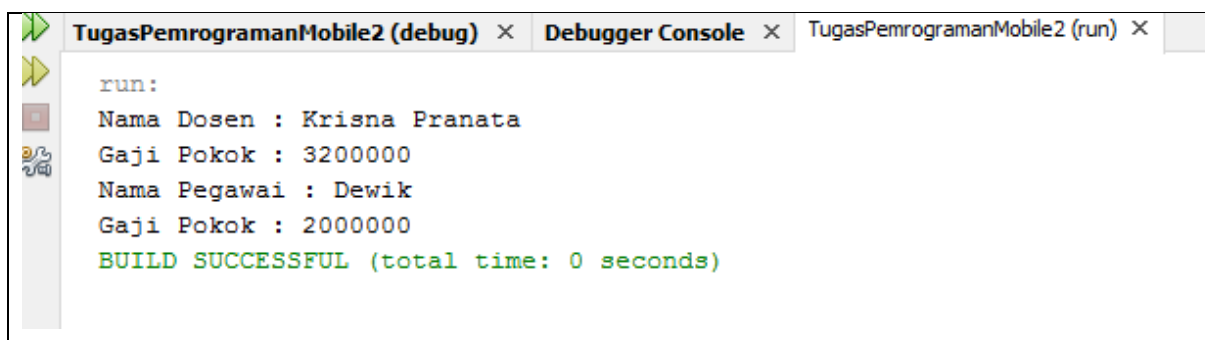
<pre>1 package tugasprogramanmobile2; 2 3 public class Pegawai { 4     private String nama; 5 6     public Pegawai() { 7     } 8 9     public Pegawai (String nama) { 10         this.nama = nama; 11     } 12     public String getNama() { 13         return nama; 14     } 15 16     public void setNama(String nama) { 17         this.nama = nama; 18     } 19 20     public int getGajiPokok() { 21         return 2000000; 22     } 23 }</pre>	<pre>1 package tugasprogramanmobile2; 2 3 public class Dosen extends Pegawai { 4     private int jumlahSKS; 5     private Account account; 6     private String addAccount; 7     private final int hargaPerSKS = 100000; 8 9     public Dosen() { 10     } 11 12     public Dosen(String nama) { 13         super(nama); 14     } 15 16     public void setJumlahSKS(int SKS) { 17         this.jumlahSKS = SKS; 18     } 19 20     public void addAccount(Account account) { 21         this.account = account; 22     } 23 24     public String getAddAccount() { 25         return account.getNama(); 26     } 27 28     @Override 29     public int getGajiPokok() { 30         int total_gaji = jumlahSKS*hargaPerSKS; 31         return total_gaji+super.getGajiPokok(); 32     } 33 }</pre>
---	---

Gambar diatas merupakan gambar yang berisi kelas Pegawai dan Dosen, dimana pada kelas Pegawai terdapat method yang bernama “getGajiPokok” yang bernilai 2000000, dan pada

kelas Dosen juga terdapat method yang sama, tetapi gaji dosen berbeda dari pegawai yang lain, jadi disini kita mengubah fungsi `getGajiPokok` menggunakan `@override`, dan mengubah fungsinya di kelas Dosen.

```
1 package tugasprogramanmobile2;
2
3 public class Main {
4     public static void main(String[] args) {
5         Detail det = new Detail();
6         // det.setNama("Krisna Pranata");
7         // det.addAccount(new Account("Khryсна"));
8         // det.setMatakuliah("Pemrograman Mobile", "P3Z2");
9         // det.setJumlahSKS(12);
10        // det.print();
11
12        Dosen dosen = new Dosen();
13        dosen.setNama("Krisna Pranata");
14        dosen.setJumlahSKS(12);
15        System.out.println("Nama Dosen : " + dosen.getNama());
16        System.out.println("Gaji Pokok : " + dosen.getGajiPokok());
17
18        Pegawai pegawai = new Pegawai ();
19        pegawai.setNama("Dewik");
20        System.out.println("Nama Pegawai : " + pegawai.getNama());
21        System.out.println("Gaji Pokok : " + pegawai.getGajiPokok());
22    }
23 }
24
```

Gambar diatas merupakan class Main, pada gambar diatas terdapat 2 panggilan yang berbeda yaitu class Dosen dan class Pegawai, tujuan class main disini yaitu membuktikan bahwa method `gajiPokok` sudah diubah atau belum pada class dosen sebelumnya.



```
TugasPemrogramanMobile2 (debug) x Debugger Console x TugasPemrogramanMobile2 (run) x
run:
Nama Dosen : Krisna Pranata
Gaji Pokok : 3200000
Nama Pegawai : Dewik
Gaji Pokok : 2000000
BUILD SUCCESSFUL (total time: 0 seconds)
```

Gambar diatas merupakan gambar output dari hasil program yang tadi, terlihat pada gambar diatas terdapat 2 nama yang berbeda yaitu nama dosen dan nama pegawai dan gaji nya pun berbeda terlihat, gaji dosen lebih besar dari gaji pegawai, ini terbukti bahwa fungsi method `gajiPokok` sudah berhasil di rubah dari class Dosen karena mengeluarkan hasil output yang berbeda dengan perhitungan yang benar juga,