

## MODUL II

### INHERITANCE, POLYMORFISME, DAN VIRTUAL METHOD INVOCATION

Tujuan Pada Pertemuan ini :

1. Memahami konsep dan dapat menerapkan *Inheritance* pada pemrograman berorientasi objek.
2. Memahami konsep dan dapat menerapkan *Polymorfisme* pada pemrograman berorientasi objek.
3. Memahami konsep dan dapat menerapkan *Virtual Method Invocation* pada pemrograman berorientasi objek.

#### 5.1 Pewarisan (*Inheritance*)

##### 5.1.1 Konsep Pewarisan Pada OOP

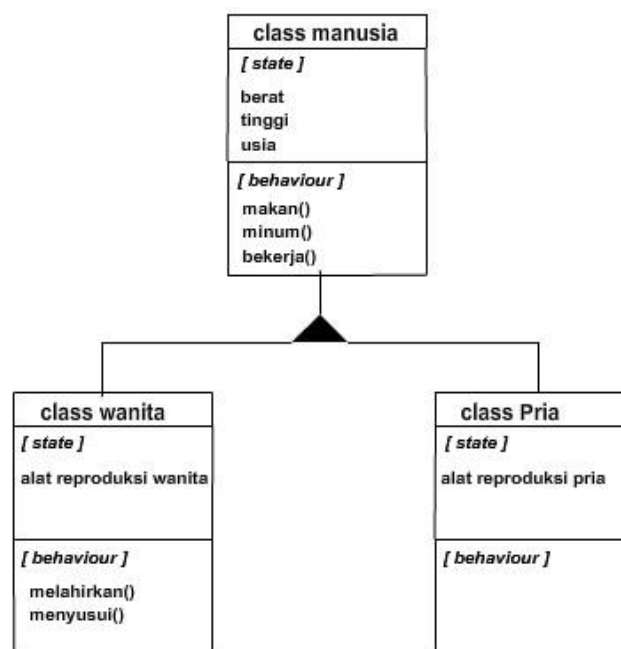
*Inheritance* merupakan suatu cara untuk menurunkan suatu kelas yang lebih umum menjadi suatu kelas yang lebih spesifik. *Inheritance* adalah salah satu ciri utama suatu bahasa program yang berorientasi pada objek, dan Java pasti menggunakannya. Java mendukung *inheritance* dengan memungkinkan satu kelas untuk bersatu dengan kelas lainnya ke dalam deklarasinya.

*Inheritance* digunakan untuk menyatukan method yang sama di setiap *class*. Sebagai contoh ketika kita membuat *class* seperti Sapi, Kambing, dan Domba. Dari ketiga *class* ini memiliki *behavior* yang sama, seperti makan, tidur, dan bergerak dan memiliki state yang sama yaitu berat badan, lebar, maupun tingginya. Karena ketiga *behavior* dan ketiga state ini dimiliki oleh class Sapi, Kambing, dan Domba maka akan lebih efektif jika membuat *class* baru untuk menyatukan *method* yang sama dan untuk meng-kategorikan *class*-nya.

Tidak hanya untuk menyatukan, *inheritance* digunakan untuk memperluas *parent class*-nya jika ingin membuat class yang spesifik. Sebagai contoh kita memiliki *class*

Mahasiswa, dan ingin membuat kategori atau cabang seperti mahasiswa dengan beasiswa, mahasiswa dengan jalur mandiri, dan mahasiswa undangan. *Subclass* yang dibuat pun bisa dari berbagai kategori.

*Class* yang mewariskan biasa disebut *parent class* atau *base class*, dan *class* yang diwariskan biasa disebut *child class* atau *subclass*. Child ini bisa memanggil dan memanipulasi seluruh *method* dan variabel pada *parent class*. Sehingga ketika mendeklarasikan sebuah objek, maka objek tersebut bisa mengakses *method* yang ada di *parent class*, maupun di *sub class*-nya.



Gambar 1 Diagram Ilustrasi Inheritance

Menurut ilustrasi diatas, ada tiga *class* berbeda dengan nama “Manusia”, “Wanita”, dan Pria. Garis diagram menunjukkan *class* “Manusia” sebagai *parent* dan *class* Wanita dan *class* Pria sebagai *subclass* dari *class* Manusia. Pada dasarnya Manusia memiliki *behavior* seperti makan, minum, dan bekerja. Manusia sendiri bisa dikategorikan menurut jenis kelaminnya sebagai Pria dan Wanita.

Tentunya *behavior* antara Pria dan Wanita memiliki kesamaan dan perbedaan, maka dari itu *behavior* yang memiliki kesamaan akan di deklarasikan pada *parent class* yaitu *class* Manusia. Untuk *behavior* yang berbeda, akan di deklarasikan pada *class* masingmasing. Sebagai contoh Wanita memiliki *behavior* melahirkan dan menyusui, dan Pria tidak memilikinya *behavior* tersebut.

Selain *behavior*, Manusia memiliki *state* seperti tinggi badan, berat badan, dan usia. Karena ketiga *state* ini memiliki kesamaan pada Pria dan Wanita, maka akan dideklarasikan pada *class* Manusia. *State* yang berbeda akan di deklarasikan di masingmasing *class* seperti alat reproduksi.

#### 5.1.2 Penerapan Konsep Pewarisan di OOP

```
1. class A {
2.
3.     int x;
4.     int y;
5.     void tampilXY(){
6.         System.out.println("Nilai x : " +x+ "\nNilai Y : "
+ y);
7.     }
8. }
9.
10.    class B extends A {
11.        int z;
12.        void jumlahXY(){
13.            System.out.println("Jumlah : " +(x+y+z));
14.        }
15.    }
16.
17.    class inheritance {
18.        public static void main(String[] args) {
19.            A superclass=new A();
20.            B subclass=new B();
21.            System.out.println("Superclass : ");
22.            superclass.x=3;
23.            superclass.y=4;
24.            superclass.tampilXY();
25.            System.out.println("Subclass : ");
26.            subclass.x=1;
```

```
27.         subclass.y=2;
28.         subclass.tampilXY();
```

```
29.         subclass.z=5;
30.         subclass.jumlahXY();
31.     }
32. }
```

Penjelasan:

1. Baris 1-8, class parent.

2. Baris 3-4, membuat variabel global untuk mendeklarasikan variabel x dan y bertipe data int.
3. Baris 5, method prosedur dengan nama tampilXY.
4. Baris 6-7, program untuk menampilkan nilai x dan nilai y.
5. Baris 9-14, class subclass dengan nama variabel B dimana class B mewarisi superclass class A.
6. Baris 10, membuat variabel global untuk mendeklarasikan variabel z bertipe data int
7. Baris 11, mencetak hasil dari penjumlahan  $x+y+z$ .
8. Baris 15-32, membuat class demoInheritance.
9. Baris 16, program utama atau main yang dipanggil ketika program dijalankan
10. Baris 17, membuat objek dari class A bernama superclass
11. Baris 18, membuat objek dari class B bernama subclass
12. Baris 19, mencetak "superclass"
13. Baris 20-22, melakukan referensi objek superclass untuk memanggil nilai  $x = 3$  dan  $y = 4$  kemudian melakukan pemanggilan prosedur tampilXY() pada class A.
14. Baris 23, mencetak "subclass"
15. Baris 25-30, melakukan referensi objek subclass untuk memanggil nilai  $x = 1$ ,  $y = 2$  dan  $z = 5$  kemudian melakukan pemanggilan prosedur pada class B yaitu jumlahXY().

Hasil output program:

```
"C:\Program Files\Java\jdk1.8.0_201\bin\java.exe" ...
superclass :
Nilai x : 3
Nilai y : 4

subclass :
Nilai x : 1
Nilai y : 2
jumlah : 8

Process finished with exit code 0
```

Gambar 2 Hasil output program Inheritance

Program ini membuat kelas untuk pewarisan untuk mengisi variabel. Memakai superclass dan subclass sehingga program akan menampilkan output seperti pada gambar diatas.

## 5.2 Polymorfisme

### 5.2.1. Polymorfime pada OOP

Polymorphism berasal dari bahasa Yunani yang berarti banyak bentuk. Dalam PBO, konsep ini memungkinkan digunakannya suatu interface yang sama untuk memerintah obyek agar melakukan aksi atau tindakan yang mungkin secara prinsip sama namun secara proses berbeda. Suatu kemampuan dari sebuah object untuk membolehkan mengambil beberapa bentuk yang berbeda agar tidak terjadi duplikasi object kita kenal sebagai polymorphism.

Polymorphism sering dikaitkan dengan penggunaan lebih dari satu metoda dengan nama sama. Penggunaan metoda dengan nama sama dapat diterapkan dengan method overloading dan method overriding. Peran polymorphism sebenarnya tidak terbatas hanya pada hal tersebut. Ada keterkaitan antara polymorphism dan inheritance (turunan).

Dalam konsep turunan, saat obyek dari subclass dikonstruksi, obyek dari superclass juga ikut dikonstruksi. Jadi setiap instance dari subclass adalah juga instance dari superclass. Apabila Anda mendeklarasikan metoda dengan parameter dari tipe superclass, Anda diperbolehkan untuk memberi argumen berupa obyek subclass yang merupakan turunan dari superclass tersebut.

Penerapan konsep polymorphism dalam pemrograman Java tidak terlepas dari konsep inheritance (turunan). Dalam konsep inheritance, ketika objek dari sub class dikonstruksi, objek dari super class juga ikut dikonstruksi. Jadi, instance dari sub class merupakan instance dari super class juga.

- Polimorfisme berarti kelas-kelas yang berbeda tetapi berasal dari satu orang tua dapat mempunyai metode yang sama tetapi cara pelaksanaannya berbeda-beda. Atau dengan kata lain, suatu fungsi akan memiliki perilaku berbeda jika dilewatkan ke kelas yang berbeda-beda.

- Polymorphism adalah suatu aksi yang memungkinkan pemrogram menyampaikan pesan tertentu keluar dari hirarki obyeknya, dimana obyek yang berbeda memberikan tanggapan/respon terhadap pesan yang sama sesuai dengan sifat masing-masing obyek.

Dalam pemrograman java, polymorphism dapat dikenali dengan adanya penggunaan lebih dari satu metode yang memiliki nama yang sama. Penggunaan metode dengan nama yang sama dapat diimplementasikan dengan method overloading atau method overriding. Atau Polymorphic dapat berarti banyak bentuk, maksudnya yaitu kita dapat menimpa (override), suatu method, yang berasal dari parent class (super class) dimana object tersebut diturunkan, sehingga memiliki kelakuan yang berbeda.

### 5.2.2. Overloading Method

Overloading merupakan salah satu metode yang memungkinkan sebuah class memiliki dua method atau lebih dengan nama yang sama, tetapi harus mempunyai parameter yang berbeda.

Yang pertama kita dapat membedakan dari jumlah parameternya (dapat berjumlah satu, dua, maupun tiga), kedua kita dapat membedakan dari tipe data parameternya (pada method pertama tipe datanya int, dan pada method lainnya tipe datanya String), dan ketiga kita dapat membedakan dari urutan dari tipe data parameternya. Method Overloading juga dikenal dengan sebutan Static Polymorphism.

Misalnya untuk melakukan operasi perhitungan terhadap angka yang diberikan, namun memiliki beberapa angka berbeda sebagai argumen, bila method ditulis seperti `j(int,int)` untuk dua parameter, dan `k(int,int,int)` untuk tiga parameter, maka akan menyulitkan kita atau programmer lainnya untuk mengerti tugas dari method tersebut, karena memiliki nama yang berbeda.

Dari metode overloading ini, yang dapat memungkinkan suatu keadaan dimana beberapa method sekaligus dapat mempunyai nama yang sama, akan tetapi mempunyai fungsionalitas yang berbeda. Overloading dapat kita ketahui dengan mudah dari ciri-ciri sebagai berikut:

1. Memiliki nama method yang sama
2. Daftar parameter pada masing-masing method harus berbeda
3. Return type yang terdapat pada method boleh sama, ataupun berbeda

Keuntungannya adalah kita tidak perlu menciptakan dan mengingat nama-nama fungsi yang berbeda untuk melakukan tugas yang sama. Contoh implementasinya adalah sebagai berikut.

```
1. class iniclass{
2.     String Variabel1;
3.     String Variabel2;
4.     void IsiParam(String parameter1) {
5.         Varibel1 = parameter1;
6.         Variebl2 = "kosong";
7.     }
```

```
8.    void IsiParam(String parameter1,String parameter2) { 9.
    Variabel1 = parameter1;
9.    Variabel2 = parameter2;
10.   }
11.   }
```

**Contoh program :**

```
1. class Lagu {
2. String pencipta;
3. String judul;
4. void IsiParam(String param1) {
5. judul = param1;
6. pencipta = "Tidak diketahui"
7. }
8. void IsiParam(String param1,String param2) {
9. judul = param1;
10.pencipta = param2;
11.}
12.void CetakKeLayar() {
13.System.out.println("Judul : " + judul + " pencipta : "
    +pencipta);
14.}
15.}
16.class demoIngerintance2 {
17.public static void main(String[] args) {
18.Lagu d,e;
19.d = new Lagu();
20.e = new Lagu();
21.d.IsiParam("Lagu 1");
22.e.IsiParam("Dan Terjadi Lagi","Peterpan");
23.  d.CetakKeLayar();
24.  e.CetakKeLayar();
25.  }
26.  }
```



Penjelasan :

1. Baris 1 sampai 11, class lagu untuk yang di dalamnya ada 2 method yang sama yaitu IsiParameter.
2. Baris 2 sampai 3, pembuatan varibel lagu dan pencipta yang type datanya string
3. Baris 4 sampai 7, membuat method prosedur mengisi judul dengan parameter1 dan pencipta dengan tidak diketahui.
4. Baris 8 sampai 16, mebuat method prosedur isiparameter sama seperti pada baris 4 tapi jumlah parameter dan argumenya berbeda
5. Baris 9, untuk mengisi judul dengan nilai paramter1.
6. Baris 10, untuk mengisi pencipta dengan nilai paramter2.
7. Baris 12 sampai 14, membuat prosedur cetakkelayar yang isinya menampilkan judul dan pencipta.
8. Baris 16 sampai 26, mainclass nya untuk memanggil dengan objek agar prosedurnya terpanggil.
9. Baris 18 – 20, pembuatan objek d dan e dengan class lagu.
10. Baris 21 sampai 22, untuk mengisi isiparameter yang d mengisi hanya dengan kalimat Lagu1, yang e mengisi untuk Judulnya dan mengisi penciptanya.
11. Baris 23 – 24 dengan memanggil prosedur cetakkelayar untuk menampilkan hasil dari mengisi objek d dan e. Yang d menampilkan lagu1, yang e menampilkan output.

Hasil Output Program :

```
"C:\Program Files\Java\jdk1.8.0_201\bin\java.exe" ...  
Judul : Lagu1  
Pencipta : Tidak diketahui  
  
Judul : dan terjadi lagi  
Pencipta : Peterpan  
  
Process finished with exit code 0
```

Gambar 3 Hasil output dari program Overloading

Pada program ini, kita menggunakan parameter untuk mengisi String pada Judul dan Pencipta lagu sehingga program akan menampilkan output seperti pada gambar diatas.

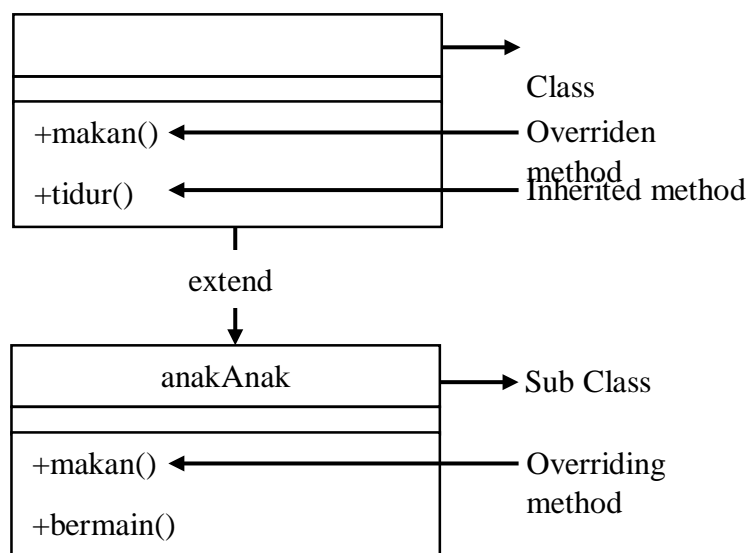
### 5.2.3. Overriding Method

Dalam Bahasa pemrograman berorientasi objek apa pun, metode Overriding adalah method yang terdapat dalam sub class. Fitur ini memungkinkan subclass atau kelas anak untuk mengimplementasikan secara spesifik metode yang ada dalam salah satu kelas super atau kelas *parent* nya.

Ketika suatu method dalam subkelas memiliki nama yang sama, parameter, dan tipe *return* yang sama sebagai method dalam super-kelasnya, maka method dalam sub kelas dikatakan menimpa method dalam superclass. Versi method yang dieksekusi akan ditentukan oleh objek yang digunakan untuk menjalankannya (merujuk pada jenis objek dan bukan tipe variabel referensi). Jika objek kelas induk digunakan untuk memanggil method, maka versi di kelas induk lah yang akan dieksekusi. Tetapi, jika objek pada subkelas yang digunakan untuk memanggil method, maka versi di kelas anak yang akan dieksekusi.

Ada beberapa aturan pada fungsi Overriding yang perlu diketahui, yaitu :

- Parameter yang terdapat pada method overriding di subclass/kelas anak harus sama dengan parameter yang terdapat pada superclass/kelas induk.
- Aturan hak akses pada fungsi overriding di subclass tidak boleh lebih ketat di bandingkan dengan hak akses method atau fungsi pada class induk.



Gambar 5.4 Ilustrasi Overriding Method

manusia

Parent

Pada gambar diatas, terdapat 2 (dua) buah class, yaitu class manusia (superclass) dan class anakAnak (sub class), class anakAnak mewarisi semua sifat pada class manusia, jadi semua attribut yang terdapat pada class manusia bisa di akses oleh class anakAnak.

Di dalam java, jika dalam suatu subclass Anda mendefinisikan sebuah method yang sama dengan yang dimiliki oleh superclass, maka method yang Anda buat dalam subclass tersebut dikatakan meng-override superclass-nya. Sehingga jika Anda mencoba memanggil method tersebut dari instance subclass yang Anda buat, maka method milik subclass-lah yang akan dipanggil, bukan lagi method milik superclass-nya. Terdapat keuntungan ketika meng-override method yaitu super class selaku pemilik overridden method, kodenya tidak perlu mengalami perubahan sama sekali, sementara itu di sisi lainnya sub class dapat mengimplementasikan kode tersebut sesuai dengan kebutuhan. Bisa dibayangkan jika suatu superclass memiliki banyak subclass. Ketika sub class – sub class tersebut perlu untuk menggunakan method dari super class, mereka dapat menggunakannya karena sub class dapat menggunakan method super class pada konsep inheritance. Dan jika mereka perlu mengimplementasikannya dengan berbeda mereka tinggal meng-override method dari super class tanpa sama sekali menyentuh kode method dari super class.

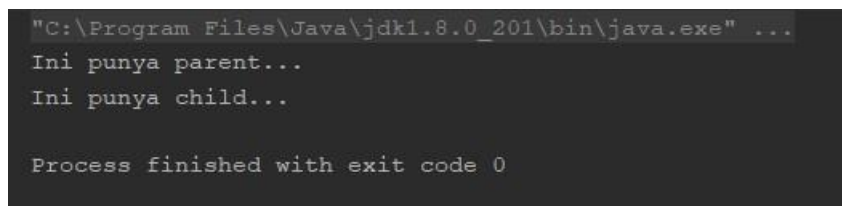
Contoh Program :

```
1.      class parent {
2.          public void showOutput() {
3.              System.out.println("Ini punya parent...");
4.          }
5.      }
6.      class child extends parent {
7.          public void showOutput() {
8.              super.showOutput();
9.              System.out.println("Ini punya child...");
10.         }
11.     }
12.     class demoInheritance {
13.         public static void main(String[] args) {
14.             child override = new child();
15.             override.showOutput();
16.         }
17.     }
```

Penjelasan :

1. Baris ke 1, penamaan program parent.
2. Baris ke 2, variabel tampilkankelayar untuk menampilkan variabel parent dan child tersebut.
3. Baris ke 3, menginput method milik kelas parent di panggil yang berguna untuk memanggil kelas parent.
4. Baris ke 6, kelas child yang diwariskan ke kelas parent.
5. Baris ke 7, variabel tampilkankelayar untuk menampilkan variabel parent dan child tersebut.
6. Baris ke 8, memanggil kelas induk dari tampilkan ke layar.
7. Baris ke 9, menginput method milik kelas child dipanggil.
8. Baris ke 13, penamaan kelas DemoInheritance.
9. Baris ke 15, membagi objek ke objek yang baru.
10. Baris ke 16, membagi objek dan tampilkan ke layar.

Hasil output program :



```
"C:\Program Files\Java\jdk1.8.0_201\bin\java.exe" ...  
Ini punya parent...  
Ini punya child...  
  
Process finished with exit code 0
```

Gambar 5 Hasil output dari program overriding

Pada program ini method yang dibuat dalam subclass akan meng-*override* superclassnya. Jika mencoba memanggil method tersebut dari subclass yang dibuat, maka method milik subclass lah yang akan dipanggil, bukan method milik superclassnya

### 5.3 *Virtual Method Invocation (VMI)*

#### 5.3.1 *Virtual Method Invocation* pada OOP

Virtual Method Invocation (VMI) bisa terjadi jika terjadi polimorfisme dan overriding. Pada saat obyek yang sudah dibuat tersebut memanggil overridden method pada parent class, kompiler Java akan melakukan invocation (pemanggilan) terhadap overriding method pada subclass, dimana yang seharusnya dipanggil adalah overridden method. Berikut contoh Virtual Method Invocation :

```
class Employee{}  
class Manager extends Employee{} ...  
Employee emp = new Manager(); emp.getDetails();
```

Pada contoh berikut objek `e` mempunyai behavior yang sesuai dengan runtime type bukan compile type. Ketika compile time `e` adalah `Employee` dan ketika runtime `e` adalah `Manager`. Maka `emp` hanya bisa mengakses variable milik `Employee` dan `emp` juga hanya bisa mengakses method milik `Manager`. Bagaimana dengan konstruktor yang dijalankan pada `Employee` `emp = new Manager()` maka pertama kali akan menjalankan konstruktor `Manager`, ketika ketemu `super()` maka akan menjalankan konstruktor `Employee` (superclass), setelah semua statement dieksekusi maka baru kemudian menjalankan konstruktor `Manager` (subclass).

Polymorphic argument adalah tipe suatu parameter yang menerima suatu nilai yang bertipe subclass-nya sehingga tetap dapat bekerja dengan baik menggunakan object dari kelas turunannya. Berikut contoh dari polymorphics argument :

```
1. class pegawai{  
2. ...  
3. }  
4. class Manager extends Pegawai{ 5....  
5. }  
6. public class Tes{  
7. public static void proses(Pegawai peg){  
8. i. ...  
9. }  
10. public static void main(String[] args){  
11. Manager man = new Manager();  
12. Proses (man); 13. }  
13. }
```

Kenapa diperlukan polymorphic arguments dalam virtual method invocation? karena dapat mengefisienkan pembuatan program misalnya `Employee` mempunyai banyak

subclass maka kita harus mendefinisikan semua method yang menangani behavior dari masing-masing subclass dengan adanya polymorphic arguments kita cukup mendefinisikan satu method saja yang bisa digunakan untuk menangani behavior semua subclass. Untuk mengetahui object yang sebenarnya dari polymorphic argument dapat digunakan operator instanceof.

Pernyataan instanceof sangat berguna untuk mengetahui tipe asal dari suatu polymorphic arguments. Untuk lebih jelasnya, misalnya dari contoh program sebelumnya kita sedikit membuat modifikasi pada class Tes ditambah sebuah class baru kurir, seperti yang tampak dibawah ini:

```
if(peg instanceof Manager){
    Manager man = (Manager) peg;
    // ...lakukan tugas-tugas manager...
}

...
```

### 5.3.2 Penggunaan VMI pada Java

```
1.      class parent {
2.          int x = 5;
3.          public void info(){
4.              System.out.println("Ini class parent");
5.          }
6.      }
7.      class child extends parent {
8.          int x = 10;
9.          public void info(){
10.             System.out.println("ini class child");
11.          }
12.      }
13.      class tes{
14.          public static void main(String[] args) {
15.              parent tes = new child();
```

```
16.          System.out.println("nilai x = " +tes.x);
```

```
17.          tes.info();
```

```
18.      }
```

```
19.  }
```

Penjelasan :

1. Baris 1 sampai 6 adalah class parent.
2. Baris 2 mendeklarasikan variabel x bertipe data int dengan value 5.
3. Baris 3 sampai 4 membuat method prosedur info dan pada baris 4 mencetak output “ini class parent”.
4. Baris 7 sampai 12 adalah class child tapi memanggil class parent menggunakan keyword extends.
5. Baris 8 mendeklarasikan variabel x bertipe data int dengan value 10.
6. Baris 9 sampai 10 membuat method prosedur info dan pada baris 10 mencetak output “ini class child”.
7. Baris 13 sampai 19 adalah class tes.
8. Baris 14 adalah method main yang dipanggil ketika program di jalankan.
9. Baris 15 membuat objek dari class parent sama dengan class child dimana program pertama kali menjalankan konstruktor class child ketika ketemu super() maka akan menjalankan konstruktor dari class parent, setelah semua statement dieksekusi maka baru menjalankan konstruktor child (subclass).
10. Baris 16 mencetak nilai dari class parent.
11. Baris 17 memanggil prosedur dari class child.

Hasil output program :

```
"C:\Program Files\Java\jdk1.8.0_201\bin\java.exe" ...  
nilai x = 5  
ini class child  
  
Process finished with exit code 0
```

Gambar 6 Hasil output dari program VMI

## 5.4 Latihan

### 5.4.1 Overloading

```
1. public class club {  
2.     public String nama;  
3.     public int tahunBerdiri;  
4.     public String stadion;  
5.     public int JuaraUcl;  
6.     public String deskripsi;  
  
7.     public club(){  
8.         this.nama = "Tidak Diketahui";  
9.         this.stadion = "Tidak Diketahui";  
10.        this.JuaraUcl = 0;  
11.        this.deskripsi = "Tidak Diketahui";  
12.    }  
  
13.    public club (String nama){  
14.        this.nama = nama;  
15.        this.tahunBerdiri = 0;  
16.        this.stadion = "Tidak Diketahui";  
17.        this.JuaraUcl = 0;  
18.        this.deskripsi = "Tidak Diketahui";  
19.    }  
  
20.    public club (String nama, String deskripsi){  
21.        this.nama = nama;  
22.        this.tahunBerdiri = 0;  
23.        this.stadion = "Tidak Diketahui";
```



```
24. this.JuaraUcl = 0;
```

```
25. this.deskripsi = deskripsi;
```

```
26. }
```

```
27. public club (String nama, int tahunBerdiri, String  
    stadion) {
```

```
28. this.nama = nama;
```

```
29. this.tahunBerdiri = tahunBerdiri;
```

```
30.this.stadion = stadion;
31.this.JuaraUcl = 0;
32.this.deskrpisi = "Tidak Diketahui";
33.}

34.public club (String nama, int tahunBerdiri, String
    stadion, int JuaraUcl, String deskripsi){

35.this.nama = nama;
36.this.tahunBerdiri = tahunBerdiri;
37.this.stadion = stadion;
38.this.JuaraUcl = JuaraUcl;
39.this.deskrpisi = deskripsi;
40.}

41.public void getTeam(){
42.System.out.println("Nama : "+nama+"\nTahun Berdiri :
    "+tahunBerdiri+"\nStadion : "+stadion+
43."\nJuara UCL : "+JuaraUcl+"\nDeskripsi : "+deskripsi);
44.}
45.}

46.public class demoClub {
47.public static void main(String[] args) {
48.club club1 = new club();
49.club club2 = new club("Barcelona");
50.club club3 = new club("Chelsea","Club diLiga Inggris");
51.club club4 = new club("Liverpool",1997,"GKTAU");
52.club club5 = new club("Indonesia",1945,"Gelora Bung
    Karno",100,"Merdeka");

53.club1.getTeam();
54.System.out.println();
```

```
55. club2.getTeam();  
56. System.out.println();  
57. club3.getTeam();  
58. System.out.println();
```

```
59. club4.getTeam();  
60. System.out.println();  
61. club5.getTeam();  
62. } 63. }
```

Penjelasan :

1. Baris 1, membuat class club
2. Baris 2 sampai 6, mendeklarasikan tipe data
3. Baris 7 sampai 12, membuat constructor tanpa parameter
4. Baris 13 sampai 19, membuat constructor dengan parameter String nama
5. Baris 20 sampai 26, membuat constructor dengan parameter String nama dan String deskripsi
6. Baris 27 sampai 33, membuat constructor dengan parameter String nama, int tahun berdiri, dan String stadion
7. Baris 34 sampai 40, membuat constructor dengan parameter String nama, int tahun berdiri, String stadion, int JuaraUcl, dan String deskripsi
8. Baris 41 sampai 44, method menampilkan output
9. Baris 45, class utama dengan nama demoClub
10. Baris 47, membuat objek 1
11. Baris 48, membuat objek 2 dengan isi parameter "Barcelona"
12. Baris 49, membuat objek 3 dengan isi parameter "Chelsea", "Club di Liga Inggris"
13. Baris 50, membuat objek 4 dengan isi parameter "Liverpool", 1997, "GKTAU"
14. Baris 51, membuat objek 5 dengan isi parameter "Indonesia", 1945, "Gelora Bung Karno", 100, "Merdeka"
15. Baris 52 sampai 60, menampilkan output dari objek yang dibuat sebelumnya

Hasil output program :

```
"C:\Program Files\Java\jdk1.8.0_201\bin\java.exe" ...
Nama : Tidak Diketahui
Tahun Berdiri : 0
Stadion : Tidak Diketahui
Juara UCL : 0
Deskripsi : Tidak Diketahui

Nama : Barcelona
Tahun Berdiri : 0
Stadion : Tidak Diketahui
Juara UCL : 0
Deskripsi : Tidak Diketahui

Nama : Chelsea
Tahun Berdiri : 0
Stadion : Tidak Diketahui
Juara UCL : 0
Deskripsi : Club di Liga Inggris

Nama : Liverpool
Tahun Berdiri : 1997
Stadion : GKT AU
Juara UCL : 0
Deskripsi : Tidak Diketahui

Nama : Indonesia
Tahun Berdiri : 1945
Stadion : Gelora Bung Karno
Juara UCL : 100
Deskripsi : Merdeka
```

Gambar 7 Hasil output dari prog

Pada program diatas membuat daftar klub juara ucl. Pada program ini akan membuat kelas dan mengambil nama, tahun berdiri, stadion, juara ucl, dan deskripsi. Dengan menggunakan tipe data string dan integer untuk membuat kelas klub tersebut.

#### 5.4.2 Overriding

```
1. public class binatang {
2.     String makan;
3.     String tidur;
4.     public void tampilBinatang (String makan,String tidur){
5.         System.out.println(makan + tidur);
6.     }
7. }
8. public class burung extends binatang {
9.     String nama;
```

```
10. public burung() { }
```

```
11. public burung(String nama) {
12. this.nama = nama;
13. }
14. public void getName(){
15. System.out.println("Burung");
16. }
17. public static void terbang(){
18. System.out.println("Terbang");
19. }
20. public void tampilBinatang(){
21. getName();
22. super.tampilBinatang("Bisa makan", " Bisa Tidur");
23. terbang();
24. }
25. }
26. public class ikan extends binatang {
27. String nama;
28. public ikan(){}
29. public ikan(String nama) {
30. this.nama = nama;
31. }
32. public void getName(){
33. System.out.println("Ikan");
34. }
35. public static void berenang(){
36. System.out.println("Berenang");
37. }
38. public void tampilBinatang(){
39. getName();
40. super.tampilBinatang("Bisa Makan", " Bisa Tidur");
41. berenang();
42. }
```

```
43. }  
44. public class kucing extends binatang {  
45. String nama;  
46. public kucing(){}  

```



```
47. public kucing(String nama){
48. this.nama = nama;
49. }
50. public void getNama(){
51. System.out.println("Kucing");
52. }
53. public static void meong(){
54. System.out.println("Meong");
55. }
56. public void tampilBinatang(){
57. getNama();
58. super.tampilBinatang("Bisa Makan"," Bisa Tidur");
59. meong();
60. }
61. }
62. public class demoBinatang {
63. public static void main(String[] args) {
64. burung binatang1 = new burung();
65. binatang1.tampilBinatang();
66. System.out.println();
67. ikan binatang2 = new ikan();
68. binatang2.tampilBinatang();
69. System.out.println();
70. kucing binatang3 = new kucing();
71. binatang3.tampilBinatang();
72. }
73. }
```

Penjelasan :

1. Baris 1, membuat class binatang
2. Baris 2 dan 3, deklarasi tipe data String

3. Baris 4 sampai 6, method tampilBinatang dengan parameter String makan dan String tidur
  4. Baris 8, kelas ikan turunan dari binatang
  5. Baris 9, deklarasi tipe data String
  6. Baris 10, constructor tanpa parameter
  7. Baris 11 sampai 13, constructor dengan parameter String nama
  8. Baris 14 sampai 16, method getNama yang menampilkan "Burung"
  9. Baris 17 sampai 19, method terbang dan bersifat static
  10. Baris 20 sampai 24, method tampilBinatang yang menampilkan isi dari getNama(),terbang(), dan turunan dari parent class
  11. Baris 26 sampai 61, penjelasan sama dengan class ikan diatas namun isi dan nama class nya berbeda, tinggal menyesuaikan saja
  12. Baris 62 dan 63, class utama dengan nama demoBinatang
  13. Baris 64, membuat objek binatang1 dari class burung
  14. Baris 65, memanggil tampilBinatang dari class burung
  15. Baris 66, berpindah ke baris berikutnya
  16. Baris 67, membuat objek binatang2 dari class ikan
  17. Baris 68, memanggil tampilBinatang dari class ikan
  18. Baris 69, berpindah ke baris berikutnya
  19. Baris 70, membuat objek binatang3 dari class kucing 20. Baris 71, memanggil tampilBinatang dari class kucing
- Hasil Output Program :

```
"C:\Program Files\Java\jdk1.8.0_201\bin\java.exe" ...  
Burung  
Bisa makan Bisa Tidur  
Terbang  
  
Ikan  
Bisa Makan Bisa Tidur  
Berenang  
  
Kucing  
Bisa Makan Bisa Tidur  
Meong
```

Gambar 8 Hasil output dari program overriding diatas

Pada gambar ini membuat program tampil binatang. Didalam nya terdapat 3 jenis hewan yaitu burung, ikan dan kucing. Menggunakan turunan atau pewarisan dalam program tersebut.

#### 5.4.3 VMI (*Virtual Method Invocation*)

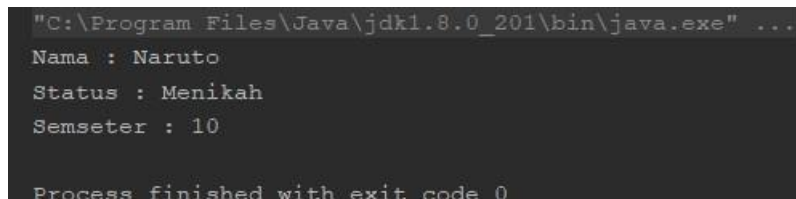
```
1. public class mahasiswa {
2. String nama;
3. String status;
4. int semester;
5. public mahasiswa(String nama, String status, int semester)
   {
6. this.nama = nama;
7. this.status = status;
8. this.semester = semester; 9. }
10. public void getStatus(){
11. System.out.println("Nama      :    "+nama+"\nStatus
    "+status+"\nSemseter : "+semester);
12. }
13. }
14. public class pacar extends mahasiswa {
15. public String namaPacar;
16. public String lamaHubungan;
17. public pacar(String nama, String status, int semester,
    String namaPacar, String lamaHubungan) {
18. super(nama, status, semester);
19. this.namaPacar = namaPacar;
20. this.lamaHubungan = lamaHubungan;
21. }
22. }
23. public class demoPacar {
24. public static void main(String[] args) {
25. pacar data = new pacar("Naruto", "Menikah", 10, "Hinata",
    "100Tahun");
26. data.getStatus();
27. }
28. }
```

:

Penjelasan :

1. Baris 1, membuat class parent mahasiswa
2. Baris 2 sampai 4, deklarasi tipe data
3. Baris 5 sampai 9, constructor mahasiswa dengan parameter String nama, String status, dan int semester
4. Baris 10 sampai 13, membuat method getStatus() dan menampilkan output
5. Baris 14, membuat class child turunan dari mahasiswa
6. Baris 15 sampai 16, deklarasi tipe data
7. Baris 18 sampai 22, constructor dengan parameter String nama, String status, int semester, String namaPacar, dan String lamaHubungan
8. Baris 23, class utama dengan nama demoPacar
9. Baris 25, membuat objek dari class pacar dengan nama data dan memberi nilai "Naruto", "Menikah", 10, "Hinata", "100Tahun"
10. Baris 27, memanggil getStatus di class mahasiswa dan menampilkan nilai dari data

Hasil Output Program :

A screenshot of a command prompt window showing the execution of a Java program. The command prompt displays the path to the Java executable and the program's output. The output shows the name 'Naruto', status 'Menikah', and semester '10'. The process finished with exit code 0.

```
"C:\Program Files\Java\jdk1.8.0_201\bin\java.exe" ...  
Nama : Naruto  
Status : Menikah  
Semester : 10  
  
Process finished with exit code 0
```

Gambar 9 Hasil output dari program VMI

Pada program ini menjelaskan tentang mahasiswa dan status mahasiswa tersebut. Didalamnya terdapat nama, status, dan lama hubungan. Sehingga mencetak status pacar.