**Bab 6**

**Inheritance, Overloading dan Overriding**

**POKOK BAHASAN**

1. Inheritance
* Pengertian dasar inheritance
* Deklarasi inheritance
* Single inheritance
* Kapan menerapkan inheritance?
* Pengaksesan member dari parent class
* Kontrol pengaksesan
* Kata kunci super
* Konstruktor tidak diwariskan
1. Overloading dan Overreading
* Overloading
* Overriding
* Aturan tentang overridden method

**TUJUAN BELAJAR**

Dengan praktikum ini mahasiswa diharapkan dapat:

1. Inheritance
* Memahami dan menerapkan konsep inheritance dalam pemrograman
* Melakukan pengontrolan akses pada pengkodean
* Memahami pengaksesan member pada parent class
* Memahami konsep package dan import
* Menggunakan kata kunci super
* Menghindari kesalahan pada pewarisan konstruktor
1. Overloading dan Overreading
* Memahami tentang overloading
* Memahami tentang overriding
* Memahami aturan tentang overridden

**Dasar Teori**

1. Inheritance

Konsep inheritance ini mengadopsi dunia riil dimana suatu entitas/obyek dapat mempunyai entitas/obyek turunan. Dengan konsep inheritance, sebuah class dapat mempunyai class turunan. Suatu class yang mempunyai class turunan dinamakan parent class atau base class. Sedangkan class turunan itu sendiri seringkali disebut subclass atau child class. Suatu subclass dapat mewarisi apa-apa yang dipunyai oleh parent class-nya, sehingga member dari suatu subclass adalah terdiri dari apa-apa yang ia punyai dan juga apa-apa yang ia warisi dari class parent-nya. Kesimpulannya, boleh dikatakan bahwa suatu subclass adalah tidak lain hanya memperluas (extend) parent class-nya.

Di dalam Java untuk mendeklarasikan suatu class sebagai subclass 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 perluasan class.

Berikut adalah contoh deklarasi inheritance:

Contoh:

public class B extends A {

...

}

Contoh diatas memberitahukan kompiler Java bahwa kita ingin meng-extend class A ke class B. Dengan kata lain, class B adalah subclass (class turunan) dari class A, sedangkan class A adalah parent class dari class B.

Java hanya memperkenankan adanya single inheritance. Konsep single inheritance hanya memperbolehkan suatu sublass mempunyai satu parent class. Dengan konsep single inheritance ini, masalah pewarisan akan dapat diamati dengan mudah.

Dalam konsep dasar inheritance dikatakan bahwa suatu subclass adalah tidak lain hanya memperluas (extend) parent class-nya. Contoh :

|  |
| --- |
| public class Pegawai {public String nama; public double gaji;} |

|  |
| --- |
| public class Manajer extends Pegawai {public String departemen;} |

Pada saat class Manajer menurunkan atau memperluas (extend) class Pegawai, maka ia mewarisi data member yang dipunyai oleh class Pegawai. Dengan demikian, class Manajer mempunyai data member yang diwarisi oleh Pegawai (nama, gaji), ditambah dengan data member yang ia punyai (departemen).

Pengaksesan member yang ada di parent class dari subclass-nya tidak jauh berbeda dengan pengaksesan member subclass itu sendiri. Contoh:

Suatu parent class dapat tidak mewariskan sebagian member-nya kepada subclass-nya. Sejauh mana suatu member dapat diwariskan ke class lain, ataupun suatu member dapat diakses dari class lain, sangat berhubungan dengan access control (kontrol pengaksesan). Di dalam java, kontrol pengaksesan dapat digambarkan dalam tabel berikut ini:

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

Kata kunci *super* dipakai untuk merujuk pada member dari parent class, sebagaimana kata kunci *this* yang dipakai untuk merujuk pada member dari class itu sendiri. Adapun format penulisannya adalah sebagai berikut:

* super.data\_member → merujuk pada data member pada parent
* class super.function\_member() → merujuk pada function member pada parent
* class super() → merujuk pada konstruktor pada parent class

Contoh:

public class Siswa {

private int nrp;

public setNrp(int nrp) {

this.nrp=nrp;

}

}

1. Overloading dan Overriding

Overloading adalah suatu keadaan dimana beberapa method sekaligus dapat mempunyai nama yang sama, akan tetapi mempunyai fungsionalitas yang berbeda. Contoh penggunaan overloading dilihat dibawah ini:

Gambar(int t1) → 1 parameter titik, untuk menggambar titik

Gambar(int t1,int t2) → 2 parameter titik, untuk menggambar garis

Gambar(int t1,int t2,int t3) → 3 parameter titik, untuk menggambar segitiga

Gambar(int t1,int t2,int t3,int t4) → 4 parameter titik, untuk menggambar persegi empat

Overloading ini dapat terjadi pada class yang sama atau pada suatu parent class dan subclass-nya. Overloading mempunyai ciri-ciri sebagai berikut:

1. Nama method harus sama

2. Daftar parameter harus berbeda

3. Return type boleh sama, juga boleh berbeda

Overriding adalah suatu keadaan dimana method pada subclass menolak method pada parent class-nya. Overriding mempunyai ciri-ciri sebagai berikut :

1. Nama method harus sama

2. Daftar parameter harus sama

3. Return type harus sama

Berikut ini contoh terjadinya overriding dimana method Info() pada class Child meng-override method Info() pada class parent:

class Parent {

public void Info() {

System.out.println(“Ini class Parent”);

}

}

class Child extends Parent {

public void Info() {

System.out.println(“Ini class Child”);

}

}

 Method yang terkena override (overridden method) diharuskan tidak boleh mempunyai modifier yang lebih luas aksesnya dari method yang meng-override (overriding method).

**Percobaan**

**Percobaan 1 : Menggunakan kata kunci *super***

Berikut ini listing penggunaan kata kunci super.

class Parent {

public int x = 5;

}

class Child extends Parent {

public int x = 10;

public void Info(int x) {

System.out.println("Nilai x sebagai parameter = " + x);

System.out.println("Data member x di class Child = " + this.x);

System.out.println("Data member x di class Parent = " +

super.x);

}

}

public class NilaiX {

public static void main(String args[]) { Child tes = new Child(); tes.Info(20);

}

}

Ketika program tersebut dijalankan, akan tampak hasil seperti dibawah ini :

Nilai x sebagai parameter = 20

Data member x di class Child = 10

Data member x di class Parent = 5

**Percobaan 2 : Kontrol pengaksesan**

Buatlah class Pegawai seperti dibawah ini:

public class Pegawai { private String nama; public double gaji;

}

Kemudian buatlah class Manajer seperti ini dibawah ini.

public class Manajer extends Pegawai {

public String departemen;

public void IsiData(String n, String d) {

nama=n;

departemen=d;

}

}

Sekarang cobalah untuk mengkompilasi class Manajer diatas. Apa yang terjadi?.

Ini membuktikan bahwa class Manajer tidak mewarisi data member nama dari parent class-nya (Pegawai).

**Percobaan 3 : Konstruktor tidak diwariskan**

Buatlah class kosong bernama Parent seperti dibawah:

public class Parent {

}

Buatlah class Child yang menurunkan class Parent seperti dibawah ini:

public class Child extends Parent {

int x;

public Child() {

x = 5;

super();

}

}

Lakukan kompilasi pada Child diatas. Apa yang terjadi?. Pasti disana terjadi error. Sekarang ubahlah sedikit class Child diatas seperti dibawah ini:

public class Child extends Parent {

int x;

public Child() {

super();

x = 5;

}

}

Setelah dikompilasi, anda tidak mendapatkan error sebagaimana yang sebelumnya. Ini yang harus kita perhatikan bahwa untuk pemanggilan konstruktor parent class, kita harus melakukan pemanggilan tersebut di baris pertama pada konstruktor subclass.

**Perbobaan 4 Melakukan overloading pada method**

Tulislah listing program berikut ini dan amati yang terjadi pada saat terjadinya overloading pada method.

import java.awt.Point;

public class Segiempat {

int x1 = 0;

int y1 = 0;

int x2 = 0;

int y2 = 0;

public void buatSegiempat(int x1, int y1, int x2, int y2) {

this.x1 = x1;

this.y1 = y1;

this.x2 = x2;

this.y2 = y2;

}

public void buatSegiempat(Point topLeft, Point bottomRight) {

x1 = topLeft.x;

y1 = topLeft.y;

x2 = bottomRight.x;

y2 = bottomRight.y;

}

public void buatSegiempat(Point topLeft, int w, int h) {

x1 = topLeft.x;

y1 = topLeft.y;

x2 = (x1 + w);

y2 = (y1 + h);

}

void cetakSegiempat(){

System.out.print("Segiempat: <" + x1 + ", " + y1);

System.out.println(", " + x2 + ", " + y2 + ">");

}

public static void main(String[] arguments) {

Segiempat rect = new Segiempat();

System.out.println("Buat segiempat dengan koordinat (25,25)

dan (50,50)");

rect.buatSegiempat(25, 25, 50, 50);

rect.cetakSegiempat();

System.out.println();

System.out.println("Buat segiempat dengan point (10,10) dan point (20,20):");

rect.buatSegiempat(new Point(10,10), new Point(20,20));

rect.cetakSegiempat();

System.out.println();

System.out.print("Buat segiempat dengan 1 point (10,10), koodinat (50,50)");

rect.buatSegiempat(new Point(10,10), 50, 50);

rect.cetakSegiempat();

}

}

Ketika program tersebut dijalankan, akan tampak hasil seperti dibawah ini :

Buat segiempat dengan koordinat (25,25) dan (50,50) Segiempat: <25, 25, 50, 50>

Buat segiempat dengan point (10,10) dan point (20,20): Segiempat: <10, 10, 20, 20>

Buat segiempat dengan 1 point (10,10), koodinat (50,50)Segiempat:

<10, 10, 60, 60>

**Latihan**

**Latihan 1 Mengimplementasikan UML class diagram dalam program untuk package perbankan**

**perbankan**

Tabungan

PenyimpananUang

- tingkatBunga : double

+ PenyimpananUang(saldo : int, tingkatBunga : double)

+ cekUang() : double

Ubahlah mode akses atribut saldo pada Tabungan menjadi protected. Lalu Transformasikan class diagram diatas ke dalam bentuk program?. Tulislah listing program berikut ini sebagai pengetesan.

import perbankan.\*;

public class TesLatihan {

public static void main(String args[]) {

PenyimpananUang tabungan=new PenyimpananUang(5000,8.5/100);

System.out.println("Uang yang ditabung : 5000"); System.out.println("Tingkat bunga sekarang : 8.5%"); System.out.println("Total uang anda sekarang : " +

tabungan.cekUang());

}

}

Lakukan kompilasi pada program diatas dan jalankan. Jika tampilan di layar tampak seperti dibawah ini, maka program anda sudah benar. Jika tidak sama, benahi kembali program anda dan lakukan hal yang sama seperti diatas.

Uang yang ditabung : 5000

Tingkat bunga sekarang : 8.5% Total uang anda sekarang : 5425.0

**Latihan 2 Mengimplementasikan UML class diagram dalam program**

Orang

# nama : String

# umur : int

+ Orang(nama : String)

+ Orang(nama : String, umur : int)

Dosen

- nip : int

+ Dosen(nama : String)

+ Dosen(nama : String, nip : int)

+ Dosen(nama : String, nip : int, umur : int)

+ Info()

Transformasikan class diagram diatas ke dalam bentuk program?. Tulislah listing program berikut ini sebagai pengetesan.

public class TesLatihan {

public static void main(String args[]) {

System.out.println("Masukkan identitas dosen 1 : Agus");

Dosen dosen1=new Dosen("Agus");

System.out.println("Masukkan identitas dosen 2 : Budi, NIP. 1458");

Dosen dosen2=new Dosen("Budi", 1458); System.out.println("Masukkan identitas dosen 3 : Iwan,

NIP. 1215, umur 47"); Dosen dosen3=new Dosen("Iwan", 1215, 47); System.out.println();

dosen1.Info();

System.out.println();

dosen2.Info();

System.out.println();

dosen3.Info();

}

}

Lakukan kompilasi pada program diatas dan jalankan. Jika tampilan di layar tampak seperti dibawah ini, maka program anda sudah benar. Jika tidak sama, benahi kembali program anda dan lakukan hal yang sama seperti diatas.

Masukkan identitas dosen 1 : Agus

Masukkan identitas dosen 2 : Budi, NIP. 1458

Masukkan identitas dosen 3 : Iwan, NIP. 1215, umur 47

Nama : Agus

NIP : -

Umur : -

Nama : Budi

NIP : 1458

Umur : -

Nama : Iwan

NIP : 1215

Umur : 47

**Tugas**

**Mengimplementasikan UML class diagram dalam program untuk package perbankan**

**perbankan**

Tabungan

PengambilanUang

- proteksi : int

+ PengambilanUang(saldo : int)

+ PengambilanUang(saldo : int, tingkatBunga : int)

+ getSaldo() : int

+ ambilUang(jumlah : int) : boolean

Transformasikan class diagram diatas ke dalam bentuk program?. Tulislah listing program berikut ini sebagai pengetesan.

import perbankan.\*;

public class TesTugas {

public static void main(String args[]) {

PengambilanUang tabungan=new PengambilanUang(5000,1000);

System.out.println("Uang yang ditabung : 5000");

System.out.println("Uang yang diproteksi : 1000"); System.out.println("-----------------"); System.out.println("Uang yang akan diambil : 4500 " +

tabungan.ambilUang(4500)); System.out.println("Saldo sekarang : " + tabungan.getSaldo()); System.out.println("-----------------"); System.out.println("Uang yang akan diambil : 2500 " +

tabungan.ambilUang(2500));

System.out.println("Saldo sekarang : " + tabungan.getSaldo());

}

}

Lakukan kompilasi pada program diatas dan jalankan. Jika tampilan di layar tampak seperti dibawah ini, maka program anda sudah benar. Jika tidak sama, benahi

kembali program anda dan lakukan hal yang sama seperti diatas.

Uang yang ditabung : 5000

Uang yang diproteksi : 1000

-----------------

Uang yang akan diambil : 4500 false

Saldo sekarang : 5000

-----------------

Uang yang akan diambil : 2500 true

Saldo sekarang : 2500