BAB 6
KONSEP INHERITANCE

A. POKOK BAHASAN

⊕ Pengertian inheritence

⊕ Deklarasi inheritance dan Single inheritance

⊕ Penerapaan inheritance

⊕ Pengaksesan member dari parent class

⊕ Kontrol pengaksesan

⊕ Kata kunci *super*

⊕ Konstruktor tidak diwariskan

B. TUJUAN BELAJAR

Dengan praktikum ini mahasiswa diharapkan dapat:

✓ Memahami dan menerapkan konsep inheritance dalam pemrograman

✓ Melakukan pengontrolan akses pada pengkodean

✓ Memahami pengaksesan member pada parent class

✓ Menggunakan kata kunci *super*

✓ Menghindari kesalahan pada pewarisan konstruktor

C. DASAR TEORI

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. 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. 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.

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.

Ada beberapa hal yang harus diingat ketika menggunakan pemanggil constuktor super:

1. Pemanggil super() HARUS DIJADIKAN PERNYATAAN PERTAMA DALAM constructor.
2. Pemanggil super() hanya dapat digunakan dalam definisi constructor.
3. Termasuk constructor this() dan pemanggil super() TIDAK BOLEH TERJADI DALAM

constructor YANG SAMA.

Contoh:

```
public class Siswa {
private int nilai;
public setNilai(int nilai) {
this.nilai=nilai;
}
}
```

D. PERCOBAAN

- **Percobaan 1**

Menggunakan kata kunci *super*

Berikut ini listing penggunaan kata kunci super untuk membedakan atribut superclass dengan atribut subclass.

```
class Bentuk {
protected int p,l; }
class Persegi extends Bentuk {
protected int p,l;
public void setSuperP(int p){
super.p = p; }
public void setSuperL(int l){
super.l = l; }
public void setP(int p){
this.p = p; }
public void setL(int l){
this.l = l; }
```

```
public void getLuas(){
System.out.println("Luas super:"+(super.l*super.p));
System.out.println("Luas:"+ (this.l*this.p)); }
}
class PersegiTest {
public static void main(String[] args){
Persegi kotak=new Persegi();
kotak.setSuperP(5);
kotak.setSuperL(10);
kotak.setP(3);
kotak.setL(13);
kotak.getLuas();
} }
```

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

```
Luas Super : 50
Luas : 39
```

- **Percobaan 2**
  Mendefinisikan Superclass dan Subclass

Buatlah 3 class dalam project kalian kemudia beri nama dan isi sebagai berikut :

Person

```
public class Person { protected String name; protected
String address;

public Person(){

System.out.println("Inside Person:Constructor");

name = "";

address = "";

}

public Person( String name, String address) {

this.name = name;

this.address = address;

}


public String getName() {

return name;

}

public String getAddress() {

return address;

}

public void setName(String name) {

this.name = name;

}

public void setAddress(String add) {

this.address = add;

}

}
```

Student

```
public class Student extends Person{

public Student()

{

//super( "Ini Nama", "Ini Alamat");

//super();

//super.name = "Ini Nama";

System.out.println("Inside Student:Constructor");

}
}
```

InheritDemo

```
public class InheritDemo {
    public static void main(String[] args) {
        Student Wahyu = new Student();
    }
}
```

Hasil dari running program diatas adalah sebagai berikut:

```
Inside Person:Constructor
Inside Student:Constructor
```

- **Percobaan 3**
  Kontrol pengaksesan

  Cobalah listing program berikut:

```
class B extends A {
private int z;
public void getJumlah(){
System.out.println("jumlah:"+ (x+y+z));
}
public void setZ(int z){
this.z = z; } }

class A {
private int x;
private int y;
public void setX(int x){
this.x = x;

public void setY(int y){
this.y = y;
}
public void getNilai(){
System.out.println("nilai x:"+ x +" nilai y:" + y);
}
class InheritanceTest{
public static void main(String [] args)
{
A ortu = new A();
B anak = new B();
      System.out.println("superclass");
ortu.setX(10);
ortu.setY(20); ortu.getNilai();
System.out.println("sub Class");
anak.setX(5);
anak.setY(4);
anak.getNilai();
anak.setz(50);
anak.getJumlah();
}
}
```

  Sekarang cobalah untuk mengkompilasi program diatas. Apa yang terjadi? Mengapa timbul pesan kesalahan dan buatlah listing program yang benar sehingga tidak timbul pesan kesalahan tersebut.

- **Percobaan 4**

  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.