MODUL PRAKTIKUM PEROGRAMAN JAVA

Tahun Akademik 2012/2013

MODUL #7

Tujuan :

- Memahami konsep dasar objek & kelas

Materi

- Mengetahui dan memahami konsep Pemrograman Berorientasi Objek
- Memahami Proses pembuatan class
- Memahami dan membuat Proses instantiate object

7.1 Konsep Dasar

Class

Java merupakan bahasa pemrograman yang murni berorientasi pada obyek. Objek adalah semua hal yang ada dalam dunia nyata, baik konkrit maupun abstrak sehingga sebuah Class menggambarkan ciri-ciri objek secara umum.

Contoh gambaran:

- Suzuki Smash, Yamaha VegaR, Honda Supra, dan Kawasaki KazeR merupakan **objek** dari **Class sepeda motor**.
- Suzuki Smash dan objek lainnya juga memiliki kesamaan **atribut** misal (merk, tipe, berat, kapasitas bensin, tipe mesin, warna, harga). Atribut juga sering disebut sebagai state/ Property.
- **Method** untuk mengakses data pada atributnya misal, fungsi untuk menginputkan data merk, tipe, berat, serta fungsi untuk mencetak data merk, tipe, berat, dan sebagainya. Method juga sering disebut sebagai tingkah laku (behavior).

Instant of Class

Object

Objek merupakan segala sesuatu yang ada didunia ini, yaitu manusia, hewan, tumbuhan, rumah, kendaraan, dan lain sebagainya. Setiap objek dalam dunia nyata pasti memiliki 2 elemen penyusunnya, yaitu keadaan (state) dan perilaku/sifat (behaviour).

- Atribut, merupakan ciri-ciri yang melekat pada suatu objek (state).
- Method, merupakan fungsi-fungsi yang digunakan untuk memanipulasi nilai-nilai pada atribut atau untuk melakukan hal-hal yang dapat dilakukan suatu objek (behaviour).

Objek yang memiliki kesamaan atribut dan method dapat dikelompokkan menjadi sebuah Class. Dan objek-objek yang dibuat dari suatu class itulah yang disebut dengan **Instant of class**. Untuk menginstansi (membuat) objek dari class, gunakan operator **new**.

Contoh 01:

Class **SepedaMotor.java**

```
public class SepedaMotor {
    private String merk, tipe;
    private int tangki;
    private long harga;
public void setMerk(String merk) {
this.merk = merk;
}
public String getMerk(){
return merk;
}

public void setHarga(long harga){
    this.harga=harga;
}
public long getharga(){
    return harga;
}

}
```

Instant of class **Latihan01.java** → Method Main

```
public class Latihan01 {
public static void main (String[]args){
SepedaMotor motor = new SepedaMotor();
motor.setMerk("Suzuki");
motor.setHarga(10000000);
System.out.println("Motor ini bermerk :" + motor.getMerk());
System.out.println("Motor ini berharga:" + motor.getharga());
}
}
```

Keterangan :

- Sintaks membuat objek dari suatu class : namaClass **namaObjek** = **new** namaClass( )
- Untuk membedakan variabel merk pada parameter dan variable merk pada atribut dari class SepedaMotor, digunakanlah keyword " **this** ".
- Ada 2 macam, yaitu method yang mengembalikan nilai dan method yang tidak mengembalikan nilai.
- Contoh method yang mengembalikan nilai adalah method **getMerk()** dimana hasil dari method ini adalah mengembalikan nilai string dari atribut merk.
- Contoh method yang tidak mengembalikan nilai adalah method **setMerk(String merk)**, yaitu dengan ciri tipe data dari method tersebut adalah void.

Access Modifier

Access modifier adalah pengaturan hak akses class maupun method. Ada 4 akses yang tersedia, yaitu default, public, protected, private.

- ✓ Private : Attribut atau method hanya bisa diakses hanya pada kelas yang sama
- ✓ Protected: Attribut atau method bisa diakses pada hanya kelas yang sama dan turunannya
- ✓ Public: Attribut atau method bisa diakses pada semua kelas.
- ✓ Default: Attribut atau method bisa diakses hanya pada kelas yang sama dan dalam package yang sama

Contoh 02:

Class **balok.java** dan Class Method Main **TestBalok.java**

```
public class balok {
// Atribute atau Property
double panjang;
double lebar;
double tinggi;
double volume;
//membuat method
void hitung() {
volume = panjang * lebar * tinggi;
System.out.println("isi balok = " + volume + "cm3");
}
}
```

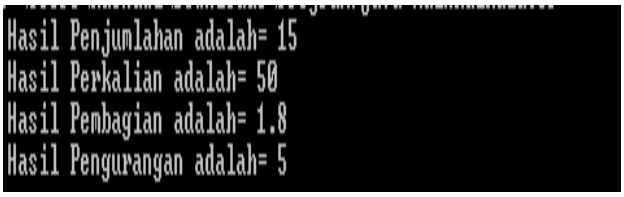
```
class TestBalok{
public static void main(String args[]){
// instansiasi
balok a = new balok();
balok b = new balok();
// Memberikan nilai pada Property a
a.panjang = 5;
a.lebar = 10;
a.tinggi = 7;
System.out.println("Balok A");
System.out.println("====================");
System.out.println("Panjang = " + a.panjang + "cm");
System.out.println("Lebar = " + a.lebar + "cm");
System.out.println("Tinggi = " + a.tinggi + "cm");
//memanggil method hitung
a.hitung();
// Memberikan nilai pada Property a
b.panjang = 10;
b.lebar = 20;
b.tinggi = 15;
System.out.println("Balok B");
System.out.println("====================");
System.out.println("Panjang = " + b.panjang + "cm");
System.out.println("Lebar = " + b.lebar + "cm");
System.out.println("Tinggi = " + b.tinggi + "cm");

//memanggil method hitung
b.hitung();
}
}
```

7.2 Latihan

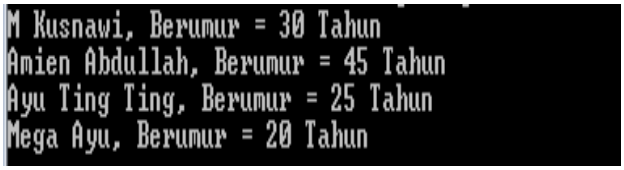
1. Membuat class Kalkulator, kemudian dalam class Kalkulator buatlah method Penjumlahan, Perkalian, Pembagian dan Pengurangan untuk dua buah nilai.

   Contoh Ouput :



2. Buatlah class Orang dengan atribut, Nama Depan, Nama Belakang dan Umur. Buat method untuk mendapatkan Nama depan dan Belakang serta Umur. Definisikan empat objek dalam class Orang.

   Contoh Ouput :



MODUL #8

Tujuan :

- Memahami konsep dasar CONSTRUCTOR & INHERITANCE

Materi :

- Mengetahui dan memahami konsep Constructor pada instantiate object
- Memahami Proses pembuatan OOP dengan Inheritance

8.1 Konsep dasar

Constructor adalah method yang secara otomatis dipanggil/dijalankan pada saat sebuah class diinstansi. Jika dalam sebuah class tidak terdapat constructor maka secara otomatis Java akan membuatkan sebuah default constructor.

Nama constructor harus sama dengan nama class dan tidak boleh memiliki tipe return value. Sama halnya dengan method, constructor dapat memiliki satu atau banyak parameter maupun tanpa parameter.

<modifier> <class_name> ([<parameter>]);

Contoh :

```
public class Employee {
private String Name;
Private double Salary;
        // Construktor
        public Employee(String Name, double Salary) {
        Salary = salary;
        Name = name;
        }
}
```

Multiple constructor adalah adanya lebih dari satu contructor untuk sebuah class. Yang membedakan antara satu constructor dengan constructor lainnya adalah pada parameternya (nama constructornya sama).

Contoh 01:

Class buku.java dan Method Main DemoBuku.java

```
public class buku {
String pengarang, judul;
buku() {
this.pengarang= "Tidak diketahui";
this.judul = "Tidak diketahui";
}
buku(String pengarang, String judul){
this.pengarang = pengarang;
this.judul= judul;
}
void cetakKeLayar() {
if(judul==null && pengarang==null)
return;
System.out.println("Judul : " +judul+", Pengarang :"+pengarang);
}

}
```

```
public class DemoBuku {
public static void main(String args[]){

    buku a = new buku("Pemograman Dasar Java ", " Iwan Sukses");
    buku b = new buku();

    a.cetakKeLayar();
    b.cetakKeLayar();


}
}
```

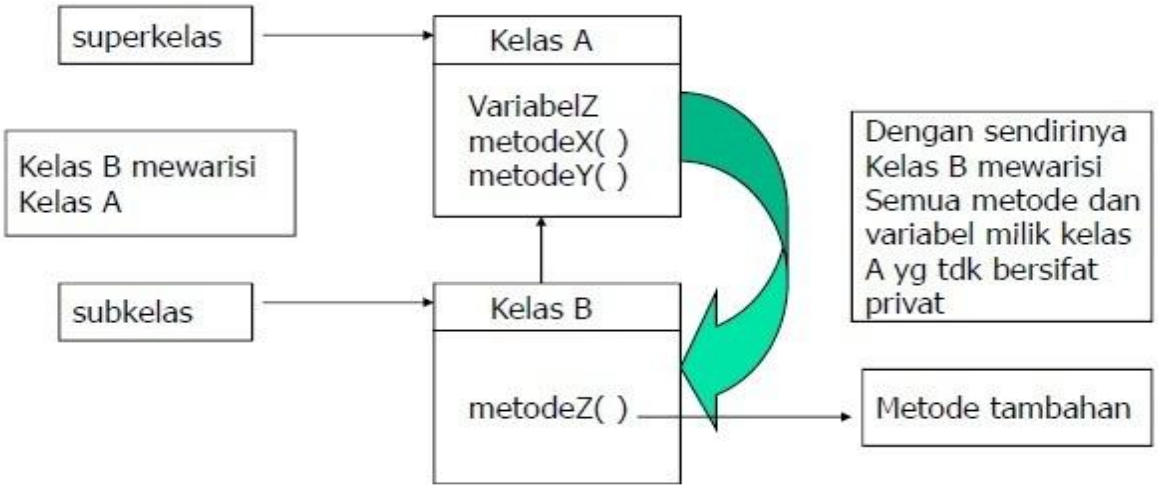
INHERITANCE

Suatu program disebut dengan pemrograman berbasis obyek (OOP) karena terdapat :

- ✓ Encapsulation (pembungkusan)
- ✓ **Inheritance** (pewarisan)
- ✓ Polymorphism (polimorfisme – perbedaan bentuk)

Inheritance, Proses pewarian data dan method dari suatu kelas kepada kelas lain. Kelas yang mewariskan disebut **super class**, sedangkan kelas yang diwariskan sering disebut **sub class**. Dengan inheritance, class yang baru (subclass) akan mirip dengan class yang lama (superclass) namun memiliki karakteristik yang baru. Contoh pewarisan : **class kendaraan** merupakan **superclass** bagi **class mobil**, **truk** dan **bis**. Penerapan pewarisan dengan menggunakan keyword **extends**.

```
public class namaSubclass extends namaSuperclass {
.....
}
```



Gambar 1. Mekanisme Inheritance

Contoh 02:

Class A.java dan Method Main DemoA.java

```
public class A {
int x,y;

void TampilkanNilaiXY(){
    System.out.println("Nilai X :"+x+", Y :"+y);
}
}

class B extends A{
    int z;

    void TampilkanJumlah(){
        System.out.println(" Jumlah :"+(x+y+z));
    }
}
```

```
public class DemoA {
public static void main(String[] args)
    {
    A superOB = new A();
    B subOB = new B();

    System.out.println("SuperClass");
    superOB.x=10;
    superOB.y=20;
    superOB.TampilkanNilaiXY();
    System.out.println("======================");
    System.out.println("SubClass");
    subOB.x=5;
    subOB.y=4;
    subOB.TampilkanNilaiXY();
    //Menambah nilai yang hanya pada subclass B
    subOB.z=50;
    subOB.TampilkanJumlah();

}
}
```

Catatan :

- suatu atribut jika dideklarasikan sebagai private pada superclass, maka tidak akan bisa diakses oleh class turunannya (subclass).
- Suatu atribut yang dideklarasikan sebagai public dan protected tetap dapat diakses oleh class turunannya.

Keyword "super"

super digunakan oleh subclass untuk memanggil constructor atau method yang ada pada superclass-nya. atau dengan kata lain jika mendeklarasikan suatu property atau method dari subclass dengan nama yang sama dengan yang dimiliki oleh superclass.

Contoh 03:

Class SuperA.java dan Method Main DemoSuperClass.java

```
public class SuperA {
int x,y;
}
class SubB extends SuperA{
    // Perhatikan pendekalarsian property x dan y
    int x,y;
    void setXYSuperClass(int x, int y)
    {
        super.x=x;
        super.y=y;
    }
    void setXYSubClass(int x, int y)
    {
        this.x=x;
        this.y=y;
    }

    void displaySuperClass()
    {
        System.out.println(" Nilai X dan Y dari Super Class: "+super.x+" dan "+super.y);
    }

    void displaySubClass()
    {
        System.out.println(" Nilai X dan Y dari SubClass :"+ x +" dan "+y);
    }
}
```

```
public class DemoSuperClass {
public static void main(String[] args){
     SubB x = new SubB();
     x.setXYSubClass(10, 20);
     x.setXYSuperClass(30, 40);

     x.displaySubClass();
     x.displaySuperClass();
}
}
```

8.2 Latihan

1. Ketikan code program, dibawa ini :

```
public class PegawaiPNS {
        String nipPegawai;  String namaPegawai;
        String jenisKelaminPegawai = "Tidak diketahui";
        int usiaPegawai; double gajiPegawai=0;
public PegawaiPNS(String nip, String nama, int usia) {
                nipPegawai=nip;
                namaPegawai=nama;
                usiaPegawai=usia;
        }
        public String getNip() {
                return nipPegawai;
        }
        public String getNama() {
                return namaPegawai;
        }
        public String getJenisKelamin() {
                return jenisKelaminPegawai;
        }
        public int getUsia() {
                return usiaPegawai;
        }
        public double getGaji() {
                return gajiPegawai;
        }
        public void setNip(String nip) {
                        nipPegawai=nip;
        }
```

Lanjutan :

```
public void setNama(String nama) {
                        namaPegawai=nama;
        }
        public void setJenisKelamin(String jenisKelamin) {
                        jenisKelaminPegawai=jenisKelamin;
        }
        public void setUsia(int usia) {
                        usiaPegawai=usia;
        }
        public void setGaji(double gaji) {
                        gajiPegawai=gaji;
        }
        public void infoPegawai() {
                System.out.println("\nInfo Pegawai");
                System.out.println("Nip\t\t: " + getNip());
                System.out.println("Nama\t\t: " + getNama());
                System.out.println("Jenis Kelamin\t: " + getJenisKelamin());
                System.out.println("Usia\t\t: " + getUsia());
                System.out.println("Gaji\t\t: " + getGaji());
} }
```

Buatlah class method main nya. Contoh outputnya :

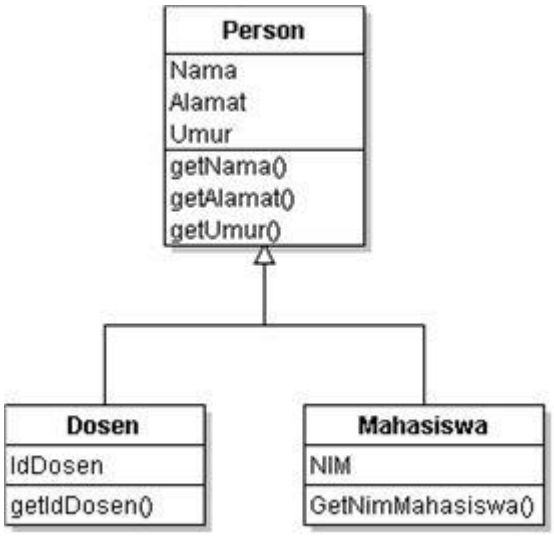
```
Info Pegawai
Nip             : 001
Nama            : Ahmad Munawari
Jenis Kelamin   : Tidak diketahui
Usia            : 26
Gaji            : 0.0

Info Pegawai
Nip             : 002
Nama            : Iman Farikhin
Jenis Kelamin   : Tidak diketahui
Usia            : 24
Gaji            : 0.0
----------------------------

Info Pegawai
Nip             : 001
Nama            : Ahmad Munawari, S.Ag
Jenis Kelamin   : Laki-Laki
Usia            : 35
Gaji            : 1500000.0

Info Pegawai
Nip             : 005
Nama            : Drs. H. Iman Farikhin, MM
Jenis Kelamin   : Tidak diketahui
Usia            : 30
Gaji            : 3000000.0
```

2. Perhatikan Gambar berikut ini, buatlah code program dengan **Inheritance**.



SuperClass : Person

SubClass : Dosen dan Mahasiswa

MODUL #9

Tujuan :

- Memahami konsep dasar OOP : ENCAPSULATION dan POLYMORPHISM

Materi :

- Mengetahui dan implementasi penyembunyian informasi (Information hiding) dan penggunakan access modifier
- Implementasi Method Overriding dan abstract class

9.1 Konsep Dasar

ENCAPSULATION

Merupakan implementasi penyembunyian informasi (Information hiding), dengan tujuan untuk menyembunyikan informasi data (field) objeck sehingga tidak terlihat dari luar(Tidak dapat diakses sembarangan). Penerapan encapsulation dapat dilakukan pada class, field ataupun metode, dengam menggunakan access modifier yang terdiri dari private, public dan protected.

Enkapsulasi mempunyai dua hal mendasar, yaitu :

1. Information hiding (menyembunyikan informasi)
   Yaitu cara memberikan hak akses private pada informasi tersebut.
2. Menambahkan method untuk mengakses informasi tersebut. Misalnya :
   - setX() : untuk memberikan nilai baru pada informasi
   - getX() : untuk mendapatkan informasi.

Access modifier adalah pengaturan hak akses class maupun method. Ada 4 akses yang tersedia, yaitu default, public, protected, private.

- Private : Attribut atau method hanya bisa diakses hanya pada kelas yang sama
- Protected: Attribut atau method bisa diakses pada hanya kelas yang sama dan turunannya
- Public: Attribut atau method bisa diakses pada semua kelas.
- Default: Attribut atau method bisa diakses hanya pada kelas yang sama dan dalam package yang sama

Contoh analoginya pada obyek Roti, obyek ini mempunyai method Pembuatan Roti. Jika kita ingin memakan roti, tentu kita tidak perlu tahu bagaimana cara membuatnya.

Contoh 01:

Class PersegiPanjang.java dan Method main DemoEncap.java

```java
public class PersegiPanjang {

private double panjang; // attribute yg di hide
private double lebar; // attribute yg di hide
private double tinggi; // attribute yg di hide

public PersegiPanjang() {
panjang = 0;
lebar = 0;
}
private double luas(double p, double l){ // di encap
return p*l;
}
public void setPanjang(double panjang) {
    this.panjang = panjang;
}
public void setLebar(double lebar) {
this.lebar = lebar;
}
public double getPanjang() {
return panjang;
}
public double getLebar() {
return lebar;
}
public double getLuas(){
return luas(panjang, lebar);
}
```

```java
public class DemoEncap {
public static void main(String[] srgs) {
PersegiPanjang pp = new PersegiPanjang();
//pp.panjang=12; tidak bisa langsung diakses
//pp.lebar=23; tidak bisa langsung diakses
pp.setPanjang(10);
pp.setLebar(20);
System.out.println("Panjang : "+ pp.getPanjang());
System.out.println("Lebar : "+ pp.getLebar());
System.out.println("Luas : "+ pp.getLuas());
}
}
```

POLYMORPHISM

Polymorphism mempunyai makna sesuatu yang memiliki banyak bentuk, yaitu memiliki nama sama, tetapi memiliki kelakuan (behaviour) yang berbeda. Polymorphism hanya berlaku pada method dan tidak berlaku untuk atribut.

Contoh 02 :

Class EkspresiWajah.java dan Method main MainEkspresiWajah.java

```
public class EkspresiWajah {

    public String respons() {
    return("Perhatikan ekspresi wajah saya");
}
}
class Gembira extends EkspresiWajah{
    public String respons() {
        return("ha ha ha..");
    }
}
class Sedih extends EkspresiWajah{
     public String respons() {
      return("hik hik ngeee ngeee ngeee..");
}
}
class Marah extends EkspresiWajah{
    public String respons() {
        return(" Hai kurang ajar loe");
    }
}
```

```
public class MainEkspresiWajah {

public static void main(String args[]) {
    EkspresiWajah objEkspresi = new EkspresiWajah();
    Gembira objGembira = new Gembira();
    Sedih objSedih = new Sedih();
    Marah objMarah = new Marah();
    EkspresiWajah[] arrEkspresi = new EkspresiWajah[4];
    arrEkspresi[0] = objEkspresi;
    arrEkspresi[1] = objGembira;
    arrEkspresi[2] = objSedih;
    arrEkspresi[3] = objMarah;
    System.out.println("Mengambil dari SuperClass: "+arrEkspresi[0].respons());
    System.out.println("Ekspresi Gembira: "+arrEkspresi[1].respons());
    System.out.println("Ekspresi Sedih: "+arrEkspresi[2].respons());
    System.out.println("Ekspresi Marah: "+arrEkspresi[3].respons());
}
}
```

Method Overriding

Overriding method adalah kemampuan dari subclass untuk memodifikasi method dari superclass-nya, yaitu dengan cara menumpuk (mendefinisikan kembali) method superclass-nya.

Contoh overriding method dapat dilihat pada class-class turunan dari class Class **EkspresiWajah** yang mendefinisikan kembali method **respon( )** dari class induknya.

Abstract Class

Abstract class adalah class yang terletak pada posisi tertinggi dari hierarki class dan digunakan sebagai acuan (superclass) bagi penurunan class-class lainnya.

Aturan-aturan dalam penggunaan class abstrak sebagai berikut :

1. Method yang tidak memiliki implementasi pada suatu class harus dideklarasikan sebagai abstrak.
2. Tidak dapat membuat instance dari class abstrak, tetapi harus diturunkan terlebih dahulu. Dimana class turunannya harus mengoverride semua method abstrak dari superclassnya dan membuat implementasinya.

Contoh 03 :

```
public abstract class Big_cat {

    protected static String name;
    protected abstract void eat();
}
class Lion extends Big_cat {
    public Lion ( String nameLion){
       Lion.name=nameLion;
    }

public void eat(){  // Method implementasi dari superclass Big_cat
    System.out.println(" Lion can eat meat");
}
}

class Tiger extends Big_cat{
    public Tiger(String nameTiger){
        Tiger.name=nameTiger;
    }
    public void eat(){// Method implementasi dari superclass Big_cat
        System.out.println(" Tiger can eat mear and drink milk");
    }
}

class Cat extends Big_cat{
    public Cat(String nameCat){
       Cat.name=nameCat;
    }
    public void eat(){// Method implementasi dari superclass Big_cat
        System.out.println(" Cat can eat meat, drink mlik and sometimes");
```

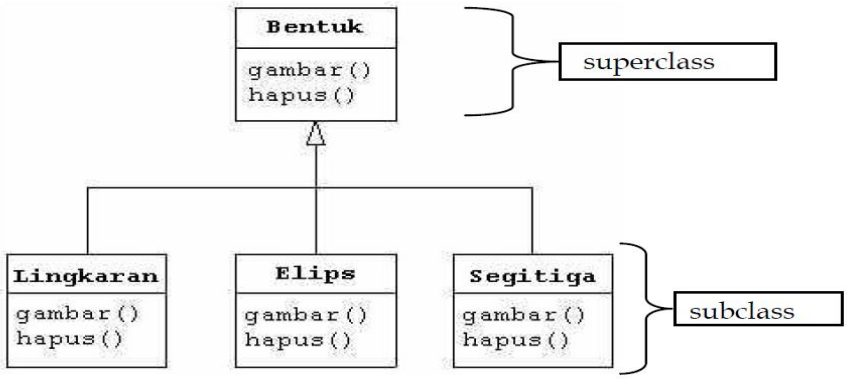
```
public class TestBig_Cat {

    public static void main(String[] args){
        Lion myLion =new Lion (" Simba");
        System.out.println(" My Lion Is"+ Lion.name );
        myLion.eat();
        Tiger myTiger =new Tiger ("Harimau");
        System.out.println(" My Tiger Is"+ Tiger.name );
        myTiger.eat();
        Cat myCat =new Cat ("Kucingku Imut");
        System.out.println(" My Lion Is"+ Cat.name );
        myCat.eat();

    }
}
```

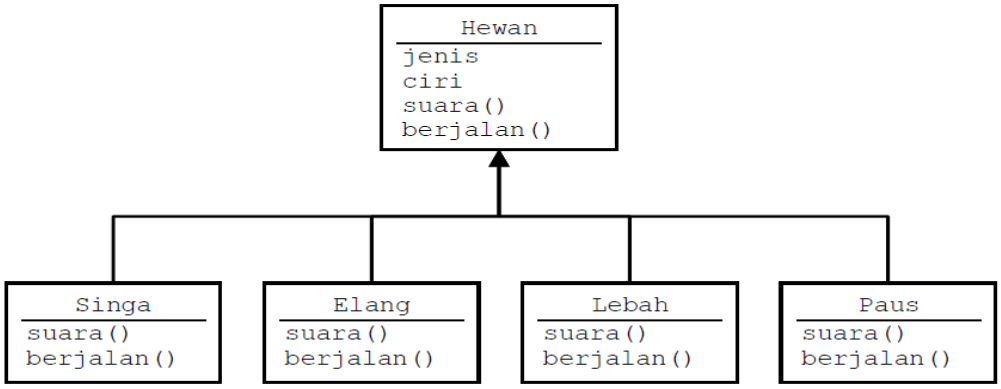
9.2 Latihan

1. Perhatikan Gambar Hirarki dibawah ini :



Buatlah program dengan teknik POLYMORPHISM.

2. Terdapat class-class hewan sebagai berikut :



Untuk penjelasan atribut dan method, jenis adalah atribut yang mengambarkan apakah termasuk hewan mamalia, serangga atau burung, ciri : ciri dari hewan tersebut, untuk method suara() : suara hewan tersebut, misalnya singa mangaom, elang crakkk dan seterusnya. Method berjalan() : cara hewan tersebut berjalan/bergerak, misalnya merangkak, terbang, berenang.

Buatlah class-class yang mengimplimentasikan gambar diatas dengan teknik POLYMORPHISM.