Modul 4
OOP-2 dan Pewarisan

A. TUJUAN

- Praktikan mampu memahami dan mengerti lebih dalam tentang OOP terutama tentang *inner class*.
- Praktikan mampu mengerti konsep dasar dan implementasi tentang pewarisan, *polymorphisme, abstract class* dan *enkapsulasi*.

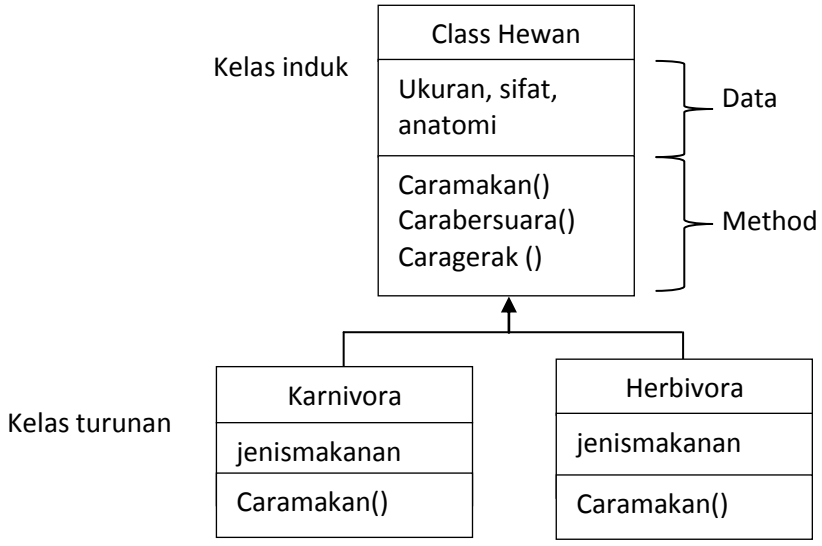
B. PERANGKAT

NetBeans IDE 7.2

C. DASAR TEORI

1. Pewarisan (*Inheritance*)

Pewarisan merupakan proses penciptaan kelas baru dengan mewarisi karakteristik kelas yang sudah ada (biasa disebut kelas induk), ditambah dengan karakteristik unik kelas baru tersebut (biasa disebut turunan). Dalam java, kelas induk ini dinamakan *superclass* dan kelas turunan dinamakan *subclass*.



Gambar 4.1 Contoh pewarisan pada kelas hewan

Hewan adalah *superclass* dari karnivora dan herbivora. Kelas turunan karnivora dan herbivora ini memiliki data dan method yang dimiliki kelas hewan.

Dalam java, format penulisan untuk membuat *subclass* adalah:

```
class namasuperclass {
// body kelas
}

class  namasubclass extends namasuperclass{
// body kelas
}
```

Contoh Program:

```
public class PersegiPanjang{
    private int panjang;
    private int lebar;

    public void setPanjang(int p){
    panjang=p;
    }
    public void setLebar(int l){
    lebar=l;
    }
    public int getPanjang(){
    return panjang;
    }
    public int getLebar(){
    return lebar;
    }
    public int Luas(){
    int luas=panjang*lebar;
    return luas;
    }
}
```

Kemudian kita buat kelas Balok yang merupakan turunan dari kelas PersegiPanjang

```
public class Balok extends PersegiPanjang{
    private int tinggi;
    public void setTinggi(int t){
    tinggi=t;
    }
    public int getTinggi(){
    return tinggi;
    }
    public int Volume(){
    int v=getPanjang()*getLebar()*tinggi;
    return v;
    }
}
```

Sedangkan untuk program utamanya:

```
public class DemoPewarisan{
   public static void main(String args[]){
         PersegiPanjang a= new PersegiPanjang();
         a.setPanjang(5);
         a.setLebar(5);
         System.out.println("");
         System.out.println("Contoh Program Pewarisan");
         System.out.println("");
         System.out.println("Superclass PersegiPanjang");
         System.out.println(" Panjang  : "+a.getPanjang());
         System.out.println(" Lebar  : "+a.getLebar());
         System.out.println(" Luas  : "+a.Luas());
         System.out.println("");

         Balok b= new Balok();
         /* kelas balok tinggal memanggil method yang ada didalam
         kelas persegi */
         b.setPanjang(4);
         b.setLebar(3);
         b.setTinggi(5);
         System.out.println("Subclass Balok");
         System.out.println(" Panjang  : "+b.getPanjang());
         System.out.println(" Lebar  : "+b.getLebar());
         System.out.println(" Tinggi  : "+b.getTinggi());
         System.out.println(" Volume  : "+b.Volume());
   }
}
```

Sehingga hasil keluaran programnya adalah:

```
Contoh Program Pewarisan

Superclass PersegiPanjang
Panjang  : 5
Lebar  : 5
Luas  : 25

Subclass Balok
Panjang  : 4
Lebar  : 3
Tinggi  : 5
Volume  : 60
```

Pewarisan menggunakan kata kunci **super** dapat dilakukan dengan format penulisan:

```
super(daftarParameter)
```

Contoh Program:

```
class Kotak(int p, int l, int t) {
   panjang = p;
   lebar = l;
   tinggi = t;
}

class  KotakPejal extends Kotak{
   private double berat;
   KotakPejal(int p, int l, int t, int b) {
   super(p, l, t);  // memanggil constructor kelas Kotak
   berat = b;
   }
}
```

2. Enkapsulasi

Enkapsulasi merupakan proses pembungkusan (encapsulation) dari suatu kelas atau biasa disebut *information hiding*. Terdapat tiga tingkat akses yang terkait dengan enkapsulasi, yaitu:

- Private

  Ketika mendeklarasikan data dan method dengan private, maka data dan method tersebut hanya dapat diakses oleh kelas yang memilikinya saja.

- Protected

  Ketika mendeklarasikan data dan method dengan protected, maka data dan method tersebut dapat diakses oleh kelas yang memilikinya dan kelas-kelas yang masih memiliki hubungan turunan.

- Public

  Data dan method yang bersifat public akan dapat diakses oleh semua bagian dalam program. Semua bagian dalam program adalah semua kelas yang memiliki hubungan turunan maupun yang tidak memiliki hubungan sama sekali

Tabel 4.1 Perbedaan Tingkat Akses Enkapsulasi

	Private	Protected	Public
Akses dalam satu kelas	bisa	bisa	bisa
Akses dalam kelas turunan	tidak	bisa	bisa
Akses dalam kelas bukan turunan	tidak	tidak	bisa

Contoh Program:

```
class musikPop {
  private String judulLagu;
// hanya dapat dikenali oleh kelas musikPop dan turunan-turunannya

   protected void setJudul(String nama) {
     judulLagu = nama;
   }
// hanya dapat dikenali oleh kelas A dan turunan-turunannya

  protected String getJudul () {
    return judulLagu;
  }
}


class musikJPop extends musikPop {
  private int tahunTerbit;
// constructor kelas B
  musikJPop(String judul, int tahun) {
//judulLagu = judul;  // SALAH, karena a tidak dikenali di sini
    setJudul(judul);  // menggunakan method setJudul()
    tahunTerbit = tahun;
  }
  public void showData() {
    // menggunakan method getJudul()
    System.out.println("Judul Lagu : " + getJudul());
    System.out.println("Tahun Terbit : " + tahunTerbit);
  }
}

class musikJazz {
  private String penyanyi;
  public void setPenyanyi(String nama) {
//setJudul('Indonesia Raya');     /* SALAH, setJudul() tidak
dikenal di sini */
    penyanyi = nama;}
   public String getPenyanyi() {
    return penyanyi;}
   public void showPenyanyi() {
 //System.out.println("Judul lagu : " + getJudul());// SALAH
    System.out.println("Penyanyi : " + penyanyi);}}


class DemoEnkapsulasi {
  public static void main(String[] args) {
// melakukan instansiasi terhadap kelas musikJPop
   musikJPop obj = new musikJPop("I Feel My Soul", 2008);
   obj.showData();
   obj.setJudul();
   System.out.println("Judul lagu : " + obj.getJudul());
  }}
```

Hasil running program di atas akan menampilkan:

```
Judul Lagu JPop : I Feel My Soul
Tahun Terbit : 2008
Judul Lagu Pop: Separuh Aku
```

3. Inner Class

Yaitu kelas yang dibuat di dalam kelas. Inner class ini dapat mengakses data dan method dari kelas yang berada di bagian luarnya, meskipun data bersifat private. Dan bagian private dari inner class tetap tidak dapat di akses oleh kelas di bagian luarnya.

Contoh Program:

```
class Luar {
  private int a;
   public void setA(int nilai) {
     a = nilai;
  }
   public int getA() {
     return a;
  }

// Mendefinisikan inner class
  class Dalam {
  private int x = 10;  // hanya dikenali oleh kelas Dalam
  public void showData() {
      System.out.println("Nilai a dari kelas Luar  : " + a);
      System.out.println("Nilai x dari kelas Dalam : " + x);
    }
}

public void showData() {
  // Baris di bawah ini SALAH, karena x tidak dikenal di sini
  //System.out.println("Nilai x dari kelas Dalam : " + x);

    Dalam obj = new Dalam();
    obj.showData();
  }
}
```

Sedangkan program utamanya:

```
class DemoInnerClass {
  public static void main(String[] args) {
Luar obj1  = new Luar();
//Dalam obj2 = new Dalam();  // SALAH

obj1.setA(5);
obj1.showData();
  }
}
```

4. Polymorfisme

Polimorfisme merupakan kemampuan suatu obyek untuk mengungkap banyak hal melalui satu cara yang sama. Misalnya kelas B, C, dan D adalah turunan kelas dari kelas A, maka kita dapat menjalankan method-method dari kelas B, C, dan D hanya dari obyek yang diinstansiasi dengan kelas A. Polimorfisme juga dapat diartikan sebagai suatu konsep pada bahasa pemrograman yang mengizinkan kelas induk untuk mendefinisikan sebuah *method general* untuk semua kelas turunannya dan selanjutnya kelas turunannya dapat memperbarui implementasi dari *method* tersebut secara lebih spesifik sesuai dengan karakteristiknya masing-masing.

5. Override

Yaitu suatu method yang ada pada kelas induk (*Superclass*) dan didefinisikan kembali oleh kelas turunan (*subclass*) dengan nama methode dan daftar parameter yang sama persis. Dan method pada kelas induknya tersebut akan disembunyikan keberadaannya. Jika kita panggil method yang sudah di-*override* dari instansiasi kelas turunannya, maka method yang dipanggil itu merupakan method dari kelas turunan tersebut, bukan method kelas induk lagi.

Perbedaan Override dengan Overload dapat dilihat dalam tabel berikut:

Tabel 4.2 Perbedaan Overload dan Override

	Overload	Override
Terdapat dalam satu kelas	ya	tidak
Terdapat dalam kelas turunannya	ya	Ya
Nama method dalam satu kelas	sama	-
Nama method pd kelas turunannya	sama	Sama
Jumlah parameter pd satu kelas	berbeda	-
Jumlah parameter pd kelas turunannya	berbeda	sama

Contoh Program:

```
public class Induk{
   public void panggilAku(){
       System.out.println("");
       System.out.println("Hallo, ini induk yang dipanggil");
   }
}

public class Anak{
   //method sama dengan method induk atau override
   public void panggilAku(){
       System.out.println("");
       System.out.println("Hallo, ini anak yang dipanggil");
   }
}

public class DemoOverride{
   public static void main(String args[]){
   Anak a= new Anak();
   a.panggilAku();
   }
}
```

Hasil keluaran ketika class DemoOverride dijalankan adalah:

```
Halo, ini anak yang dipanggil
```

6. Abstraksi

Yaitu penyembunyian kerumitan dari suatu proses. Biasa berbentuk kelas murni yang tidak boleh memiliki objek, dan satu/lebih method-methodnya yang *abstract* harus diimplementasikan (override) oleh kelas turunannya. Dan kelas abstrak ini tidak dapat diinstansiasi.

Format penulisan abstrak dalam java adalah

```
abstract class NamaKelas {
// deklarasi attribute
// definisi/prototype method
}
```

Contoh Program:

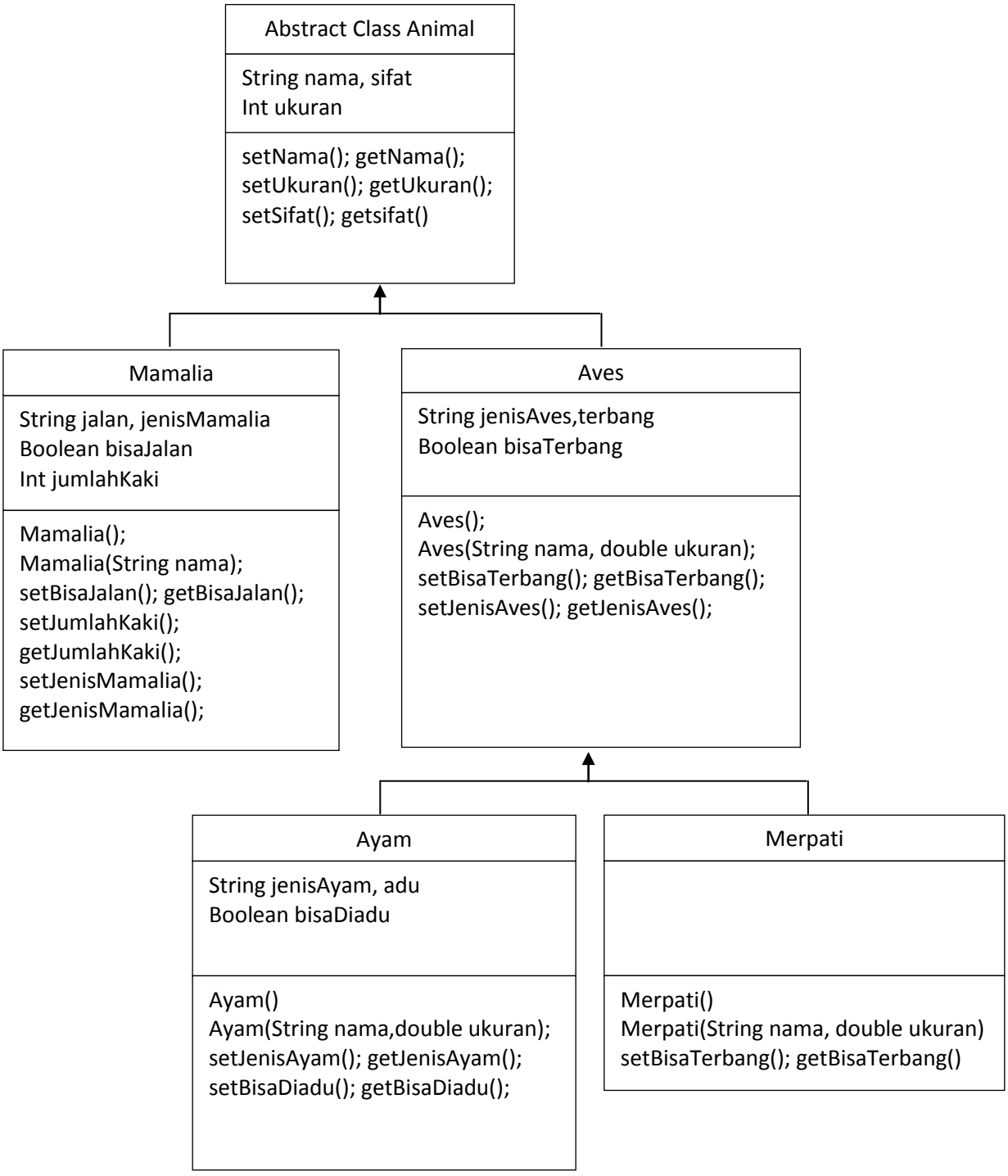
```
abstract class Hewan {
   protected String nama;
   protected int jumKaki;
   protected boolean bisaTerbang = false;
   public Hewan(String nama, int kaki, boolean terbang) {
       this.nama = nama;
       jumKaki = kaki;
       bisaTerbang = terbang;
   }
   public abstract void bersuara();
```

```
   public void lihatHewan() {
       System.out.println("");
       System.out.println("nama  : "+nama);
       System.out.println("jumlah kaki  : "+jumKaki);
       System.out.println("bisa terbang  : "+bisaTerbang);
   }
}

class Sapi extends Hewan {
   public Sapi() {
    super("sapi", 4, false);
   }
   public void bersuara() {
    System.out.println("\nmoooaahhhh,mooooaaahhh");
   }
   public static void main(String[] args) {
       Sapi s = new Sapi();
       s.lihatHewan();
       s.bersuara();
   }
}

class Perkutut extends Hewan {
   public Perkutut(){
    super("perkutut",2,true);
   }
   public void bersuara() {
    System.out.println("\ncuit, cuit, cuit");
   }
   public static void main(String[] args) {
       Perkutut p = new Perkutut();
       p.lihatHewan();
       p.bersuara();
   }
}
```

Ketika class Sapi dijalankan, maka keluarannya:

```
nama  : sapi
jumlah kaki  : 4
bisa terbang  : false

moooaahhhh,mooooaaahhh
```

Contoh UML dengan kelas turunan tingkat 2.



Gambar 4.2 Contoh UML dengan kelas turunan tingkat 2