MODUL 4
ENCAPSULATION, INHERITANCE DAN POLYMORPHISM

A. ENCAPSULATION (PENGKAPSULAN)

Encapsulation adalah proses pemaketan data bersama metode-metodenya dimana hal ini bermanfaat untuk menyembeunyikan rincian-rincian implementasi dari pemakai. Dalam sebuah objek yang mengandung variabel-variabel dan methodmethod, dapat ditentukan hak akses pada sebuah variabel atau method dari objek. Pembungkusan variabel dan method dalam sebuah objek dalam bagian yang terlindungi inilah yang disebut dengan ***enkapsulasi***.

Bagian eksternal dari sebuah objek sering disebut sebagai *interface* atau *antarmuka* terhadap objek lain. Karena objek lain harus berkomunikasi dengan obejk itu hanya melalui antarmuka maka bagian internal objek dapat dilindungi dari gangguan luar. Karena program luar tidak mengakses implementasi internal objek, maka implementasi internal dapat berubah tanpa mempengaruhi bagian-bagian program lain.

Di dalam Java, pengkapsulan dapat dilakukan dengan pembentukan kelas-kelas menggunakan keyword *class*. Sedangkan penyembunyian informasi dapat dilakukan dengan pengendalian tgerhadap pengaksesan pembentuk kelas dengan keyword-keyword untuk kendali pengaksesan *default, private, protected,* dan *public*. Penyembunyian informasi deilakukan dengan implementasi penerapan kendari menggunakan keyword *private* dan *protected* pada elemen data.

Ada 2 manfaat utama dari enkapsulasi yaitu :

1. Penyembunyian Informasi (*information hiding*)
   Ha ini mengacu kepada perlindungan terhadap implementasi obejk internal. Obejk tersebut dari interface *public* dan bagian *private* yang merupakan kombinasi data dan metode internal. Manfaat utamanya adalah bagian internal dapat berubah tanpa memepngaruhi bagian-bagian program yang lain.
2. Modularitas
   Modularitas berarti objek dapat dikelola secara independen. Karena kode sumber bagian internal objek dikelola secara terpisah dari antarmuka, maka Kita bebas melakukan modifikasi yang tidak menyebabkan masalah pada bagian-bagian lain dari sistem. Manfaat ini mempermudah mendistriibusikan objek-objek dari sistem.

Untuk lebih memahami berikut contoh Program dengan implementasi Enkapsulasi :

```
public class Kapsul{
   private double panjang; // attribute yang disembunyikan
   private double lebar;     // attribute yang disembunyikan
   private double tinggi;    // attribute yang disembunyikan

public Kapsul(){
      double panjang = 0;
      double lebar = 0;
   }
```

```
    private double luas(double p, double l) {  //attribute yang di enkapsulasi
      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);
    }
}

class Encapsulasi{
    public static void main(String[] srgs) {
      Kapsul pp = new Kapsul();
      pp.setPanjang(50);
      pp.setLebar(100);
      System.out.println("Panjang : "+ pp.getPanjang());
      System.out.println("Lebar : "+ pp.getLebar());
      System.out.println("Luas : "+ pp.getLuas());
    }
}
```

Dari program di atas dapat dilihat bahwa deklarasi variabel disembunyikan dan pada saat dilakukan proses perkalian untuk mencari *luas* dilakukan enkapsulasi. Program di atas akan menghasilkan 2 *class* yaitu : *class Kapsul* dan *class Encapsulasi*. Kedua *class* ini berbeda. Di mana *class Kapsul* merupakan kelas untuk pembentukan Objek dan pemrosesan dalam pencarian luas persegi panjang. Sedangkan untuk *class Encapsulasi* merupakan kelas untuk menampilkan hasil dari proses tersebut.

B. INHERITANCE (PEWARISAN)

Pewarisan adalah proses penciptaan kelas baru dengan mewarisi karakteristik dari kelas yang telah ada, ditambah dengan karakteristik unik dari kelas baru tersebut. Dalam hirarki kelas, jika kelas C merupakan turunan kelas B, dan kelas B merupakan turunan kelas A, maka otomatis attribute dan method kelas A juga diwariskan kelas C. Setiap ***subclass*** akan mewarisi ***state*** (variabel-variabel) dan ***behaviour*** (method-method) dari ***superclass***-nya. ***Subclass*** kemudian dapat menambahkan ***state*** dan ***behaviour*** baru yang spesifik dan dapat pula memodifikasi (***override***) ***state*** dan ***behaviour*** yang diturunkan oleh ***superclass***-nya.

Keuntungan dari inheritance adalah :

- ***Subclass*** menyediakan ***state/behaviour*** yang spesifik yang membedakannya dengan ***superclass***, hal ini akan memungkinkan programmer Java untuk menggunakan ulang ***source code*** dari ***superclass*** yang telah ada.
- Programmer Java dapat mendefinisikan ***superclass*** khusus yang bersifat generik, yang disebut ***abstract class***, untuk mendefinisikan ***class*** dengan ***behaviour*** dan ***state*** secara umum.
- Kemudahan dalam me-manage kelas yang memiliki data dan method yang sama
  Untuk memodifikasi suatu data atau method untuk semua subkelas / kelas anak, maka tidak perlu melakukan perubahan di masing-masing kelas anak melainkan hanya pada kelas induk saja.

Istilah dalam ***inheritance*** yang perlu diperhatikan :

- ***Extends***
  Keyword ini harus kita tambahkan pada definisi class yang menjadi subclass.
- ***Superclass***
  Superclass digunakan untuk menunjukkan hirarki class yang berarti class dasar dari subclass/class anak.
- ***Subclass***
  Subclass adalah class anak atau turunan secara hirarki dari superclass.

Secara umum bentuk deklarasi dalam kkonsep pewarisan adalah sebagai berikut :

```
[modifier] class namaSubKelas extend namaKelasSuper {
  // classBody
}
```

Untuk lebih memahami marilah Kita lihat contoh berikut :

```
public class Komunitas {
   String nama;
   String alamat;
   String alamat2;
   String kota;
   int umur;

   /** Defenisi kelas Komunitas */
   public Komunitas() {
          nama = "";
          alamat = "";
          alamat2 = "";
          kota = "";
          umur = 0;
   }

   public Komunitas(String newNama, String newAlamat, String newAlamat2, String newKota, int
newUmur) {
          nama = newNama;
          alamat = newAlamat;
          alamat2 = newAlamat2;
          kota = newKota;
          umur = newUmur;
   }

   public void setNama (String newNama){
          nama = newNama;
   }

   public void setAlmat (String newAlamat){
          alamat = newAlamat;
   }

   public void setAlamat2 (String newAlamat2) {
          alamat2 = newAlamat2;
   }

   public void setKota (String newKota) {
          kota = newKota;
   }

   public void setUmur (int newUmur) {
          umur = newUmur;
   }
```

```
public String getNama (){
        return nama;
   }

   public String getAlamat (){
        return alamat;
   }

   public String getAlamat2 (){
        return alamat2;
   }

   public String getKota (){
        return kota;
   }

   public int getUmur (){
        return umur;
   }

   public String toString(){
      String str =
         "Nama     : "+ nama + "\n"+
         "Alamat 1  : "+ alamat + "\n"+
         "Alamat 2  : "+ alamat2 + "\n"+
         "Kota     : "+ kota + "\n"+
         "Umur     : "+ umur + "\n";
      return str;
   }

   static void nilai(){
   Komunitas t = new Komunitas ("Sukma Murdani, S.Kom","Jalan Aru","Lubuk
Begalung","Padang",25);

      System.out.println("\nSukma Murdani sebagai Dosen Komunitas:\n");
      System.out.println(t);
   }

   public static void main(String[] args) {
      nilai();
   }
}
```

Kelas *Komunitas* di atas adalah merupakan kelas induk yang kemudian bisa diakses oleh kelas anak mana saja. Berikut salah contoh kelas anak yang mengakses kelas *Komunitas.*

```
public class Mahasiswa extends Komunitas {
   String BP;
   String strata;
   String Jurusan;

   /** Penciptaan Kelas Mahasiswa sebagai Pewarisan dari Komunitas */
   public Mahasiswa() {
          super();
          BP = "";
          strata = "";
          Jurusan = "";
   }

   public Mahasiswa (String newNama, String newAlamat, String newAlamat2, String newKota, int
newUmur, String newBP, String newStrata, String newJurusan) {
          super(newNama, newAlamat, newAlamat2, newKota, newUmur);
          BP = newBP;
          strata = newStrata;
          Jurusan = newJurusan;
   }

   public void setBP (String newBP) {
          BP  = newBP;
   }
public String getBP () {
          return BP;
   }

   public void setStrata (String newStrata) {
          strata = newStrata;
   }

   public String getStrata () {
          return strata;
   }

   public void setJurusan (String newJurusan) {
          Jurusan = newJurusan;
   }

   public String getJurusan() {
          return Jurusan;
   }
```

```
public String toString(){
      String str =
         "Nama     : "+ nama + "\n"+
         "Alamat 1  : "+ alamat + "\n"+
         "Alamat 2  : "+ alamat2 + "\n"+
         "Kota     : "+ kota + "\n"+
         "Umur     : "+ umur + "\n"+
         "NRP      : "+ BP + "\n"+
         "Strata   : "+ strata + "\n"+
         "Jurusan   : "+ Jurusan;
      return str;
   }

  static void nilai(){
      Mahasiswa t = new Mahasiswa ("James Bond","Jalan Aru","Lubuk
Begalung","Padang",20,"10007","Strata 1","Teknik Informatika");

      System.out.println("\nJames Bond sebagai Mahasiswa Komunitas:\n");
      System.out.println(t);
   }

   public static void main(String[] args) {
     nilai();
   }
}
```

Dari contoh Program di atas terlihat bahwa *class Mahasiswa* mewarisi seluruh sifat yang ada pada *class Komunitas*.

C. POLYMORPHISM

Polymorphism dapat diartikan sebagai memiliki banyak bentuk. Dua objek atau lebih dikatakan sebagai *polymorphic* bila kedua objek tersebut mempunyai antarmuka identik namun mempunyai perilaku yang berbeda. Dalam pemrograman, polimorfisme dapat diartikan sebagai modul yang memiliki nama sama, namun memiliki behaviour (tingkah laku) yang berbeda sehingga listing code implementasinya juga berbeda.

Kondisi yang harus dipenuhi supaya ***polimorfisme*** dapat diimplementasikan adalah :

- Method yang dipanggil harus melalui variabel dari basis class atau superclass.
- Method yang dipanggil harus juga menjadi method dari basis class.
- Signature method harus sama baik pada superclass maupun subclass.
- Method access attribute pada subclass tidak boleh lebih terbatas dari basis class.

Untuk dapat memahami metode *polimorfisme* marilah Kita lihat contoh Script di bawah ini :

Contoh 1 :

```
abstract class Bentuk {
protected int panjang;
protected int lebar;
public String getBentuk() {
return "Bentuk Dasar";
}
public abstract int hitungLuas();
}
class BujurSangkar extends Bentuk {
public BujurSangkar(int panjang1, int lebar1) {
this.panjang = panjang1;
this.lebar = lebar1;
}
public String getBentuk() {
return "Bentuk Bujur Sangkar";
}
public int hitungLuas() {
return panjang*lebar;
}
}
class SegiTiga extends Bentuk {
public SegiTiga(int panjang2, int lebar2) {
this.panjang = panjang2;
this.lebar = lebar2;
}
public String getBentuk() {
return "Bentuk Segitiga";
}
public int hitungLuas() {
return this.panjang*this.lebar/2;
}
}

class Polimorfisme {
public static void cetakLuasBentuk(Bentuk btk) {
System.out.println(btk.getBentuk() + " dengan luas " +btk.hitungLuas());
}
public static void main(String[] args) {
BujurSangkar bs = new BujurSangkar(10,10);
SegiTiga st = new SegiTiga(5,10);
cetakLuasBentuk(bs);
cetakLuasBentuk(st);
}
}
```

Contoh 2 :

```
class Mimik{
   public String respons() {
      return("Ini EkspresiKu, Mana EkspresiMu");
   }
}
class Senang extends Mimik{
   public String respons() {
      return("Wk...wk...wk...yiii...haaaa...");
   }
}
class Sedih extends Mimik{
   public String respons() {
      return("hiks..hiks...Aaakuu Gaalaauu...");
   }
}
class Marah extends Mimik{
   public String respons() {
      return("Wwoooiii....Gue Tabok Loe..!!");
   }
}

class Tampil{
   public static void main(String args[]) {
     Mimik objEkspresi = new Mimik();
      Senang objGembira = new Senang();
      Sedih objSedih = new Sedih();
      Marah objMarah = new Marah();

      Mimik[] arrEkspresi = new Mimik[4];
      arrEkspresi[0] = objEkspresi;
      arrEkspresi[1] = objGembira;
      arrEkspresi[2] = objSedih;
      arrEkspresi[3] = objMarah;

      System.out.println(""+arrEkspresi[0].respons());
      System.out.println("Gue lagi seneng ne "+arrEkspresi[1].respons());
      System.out.println("Huh sedih banget "+arrEkspresi[2].respons());
      System.out.println("Arrggttthh...!! "+arrEkspresi[3].respons());
   }
}
```