MATERI 3 – ENKAPSULASI

PEMROGRAMAN LANJUT S1 TI

STEVI EMA WIJAYANTI

CLASS DAN OBJEK

DEFINISI CLASS

Class merupakan suatu "*blueprint*" atau "*cetakan*" untuk menciptakan suatu instan dari *object*. Class juga bisa disebut blok-blok bangun pada program Java. Class mendefinisikan atribut/properti atau biasa disebut dengan ***field*** dan perilaku (*behavior*) atau biasa disebut dengan ***method*** dari obyek yang dibuatnya.

DEFINISI ATTRIBUTE

Attribute berlaku sebagai data, didefinisikan oleh class, individu, dan berbeda satu dengan lainnya. *Attribute* didefinisikan dalam class dengan menggunakan variabel.

DEFINISI BEHAVIOR

Behavior berlaku sebagai method (operasi). Untuk menentukan *behavior* atau perilaku dari *object* harus membuat ***methods***.

CONTOH :

Class : Manusia

Attribute : 1. Nama
2. Gender
3. Tempat tanggal lahir
4. Alamat
5. dan sebagainya.

Behavior : 1. Berjalan
2. Makan
3. Tidur
4. dan sebagainya.

ANALOGI LAIN

Bagaimana cara membuat mobil ?

Langkah 1 :

Menggambar desain mobil

Langkah 2 :

Menambahkan detail mobil, misalnya : desain pedal gas untuk maju, desain rem untuk berhenti, desain setir untuk belok kiri dan kanan, dan lain sebagainya.

Langkah 3 :

Proses pembuatan mobil dimulai

Bagaimana pedal gas pada mobil bekerja ?

1. Bagaimana proses yang terjadi di dalam mesin mobil ketika pedal gas diinjak ?
2. Apakah pengendara mobil perlu mengetahui cara kerja pedal gas secara detail ?

Pengendara mobil tentu tidak perlu mengetahui secara detail cara kerja pedal gas ketika diinjak. Pengendara cukup mengetahui aturan dan fungsi dari pedal gas tersebut, misalnya ketika pedal gas sebelah tengah diinjak maka mobil akan melaju. Hal tersebut dapat disimpulkan bahwa cara kerja pedal gas secara detail disembunyikan dari pengendara mobil.

Dari analogi diatas dapat diambil kesimpulan :

1. Mobil adalah sebuah CLASS, dimana class Mobil memiliki *attribute* dan *behavior*.
2. Sama halnya penyembunyian cara kerja pedal gas secara detail pada sebuah mobil, pada OOP penyembunyian detail pemrograman di dalam sebuah METHOD.
3. Method-method ini dibuat dan disimpan bersama-sama di dalam suatu desain yang disebut Class.

DEFINISI OBJECT

Objek adalah intansiasi (*instance*) dari class. Jika class secara umum mempresentasikan (*template*) sebuah objek, sebuah *instance* adalah representasi nyata dari class itu sendiri.

Cara pembuatan objek :

```
NamaKelas  NamaObjek = new NamaConstruktor();

Atau

NamaKelas NamaObjek;

NamaObjek = new NamaConstruktor();
```

Contoh : Dari class Mobil, dapat dibuat objek Sedan, Kijang, Truk, dan lain sebagainya.

DEFINISI FUNGSI

Fungsi adalah suatu operasi berupa fungsi-fungsi yang dapat dikerjakan oleh suatu objek atau dapat dikatakan *method* adalah perilaku dari objek.

Fungsi memiliki 4 (empat) bagian dasar, antara lain :

1. Nama fungsi
2. Tipe Objek atau tipe primitive yang dikembalikan metode
3. Daftar parameter
4. Badan atau isi fungsi

Construktor merupakan fungsi khusus yang digunakan untuk menginisialisasi objek dan masing-masing class pasti memiliki minimal satu construktor. Perbedaan fungsi biasa dengan construktor adalah nama fungsi. Construktor memiliki nama fungsi sama persis dengan nama classnya, tidak bertipe dan tidak memiliki nilai kembalian.

CONTOH IMPLEMENTASI CLASS DAN OBJEK

Contoh 1 : Class dan Method Buah

Buah.cs

```
public class Buah {
    //method 1
    public void Mangga(){
        Console.WriteLine("Ini method untuk buah mangga");
    }
    //method 2
    public void Apel(){
        Console.WriteLine("Ini method untuk buah apel");
    }
}
```

AmbilBuah.cs

```
public class AmbilBuah{
    static void Main(String [] args){
        //instance
        Buah objB = new Buah();
        b.Mangga();
        b.Apel();
    }
}
```

Contoh 2 : Program Perhitungan Volume Balok

Balok.cs

```
public class Balok{
    //declare attribute/property
    double panjang;
    double lebar;
    double tinggi;
}
```

VolumeBalok.cs

```
public class VolumeBalok{
    static void Main(String [] lat){
        double volume;

        //instance
        Balok objBl = new Balok();

        objBl.panjang = 10;
        objBl.lebar = 5;
        objBl.tinggi = 5;

        volume = objBl.panjang * objBl.lebar * objBl.tinggi;
        Console.WriteLine("Volume = "+volume+" cm3");
    }
}
```

Contoh 3 : Program Perhitungan SPP

SPP.cs

```
class SPP {
    private string nama,nim;
    private int spt,spv,sks;

    //constructor
    public SPP(String nama, String nim, int spt, int spv, int sks){
        this.nama = nama;
        this.nim = nim;
        this.spt = spt;
        this.spv = spv;
        this.sks = sks;
    }
```

```
    //method
    public int hitung(){
        return (spt+(spv*sks));
    }
}
```

DemoSPP.cs

```
class DemoSPP {
    static void Main (string [] coba){
        string nama,nim;
        int spt,spv,sks;

        //input
        Console.Write("Nama \t\t: ");
        nama = Console.ReadLine();
        Console.Write("NIM \t\t: ");
        nim = Console.ReadLine();
        Console.Write("SPP Tetap \t: ");
        spt = Int32.Parse(Console.ReadLine(););
        Console.Write("SPP Variabel \t: ");
        spv = Int32.Parse(Console.ReadLine(););
        Console.Write("SKS \t\t: ");
        sks = Int32.Parse(Console.ReadLine(););

        //proses instance + input
        SPP bayar = new SPP(nama,nim,spt,spv,sks);
```

```
        //output
        System.out.println("\n-------------- PROGRAM PERHITUNGAN SPP OTOMATIS --------------\n");
        System.out.println("Nama \t\t: " + nama);
        System.out.println("NIM \t\t: " + nim);
        System.out.println("SPP Tetap \t: " + spt);
        System.out.println("SPP Variabel \t: " + spv);
        System.out.println("SKS \t\t: " + sks);
        System.out.println("Total SPP yang harus dibayarkan : Rp " + bayar.Hitung() + ",-\n");
        System.out.println("------------------------ TERIMA KASIH ------------------------\n");

    }
}
```

ENKAPSULASI

Seperti yang diketahui enkapsulasi merupakan salah satu konsep dasar OOP (*Object Oriented Programming*). Sesuai dengan arti namanya yaitu pembungkusan, maka dalam konsep ini variable yang digunakan oleh sebuah program dideklarasikan dengan hak akses ***private***.

Jika sebelumnya pada Bahasa pemrograman C++, digunakan fungsi **set** dan **get** untuk menginputkan dan mengambil nilai dari suatu variable, seperti contoh berikut ini :

Fungsi **Set** :

```
//mendeklarasikan fungsi set dengan nama SetNama() dan bertipe Void
//method SetNama() memiliki 1 paramater "name" dan bertipe string
public void SetNama(string name){
        nama = name;
}
```

Fungsi **Get** :

```
//mendeklarasikan fungsi get dengan nama GetNama() dan bertipe String
public string GetNama(){
        //karena non-void maka harus ada nilai kembalian (return value)
        return nama;
}
```

Pada Bahasa pemrograman C# dikenal sebagai ***property*** yang dapat menggantikan fungsi set dan get pada Bahasa pemrograman C++, seperti contoh berikut ini :

```
//mendeklarasikan property sebagai pengganti fungsi set dan get
private string nama;
public string Nama {  //property untuk variable nama
        //bagian ini digunakan untuk menginputkan data/value ke variable nama
        set {
                nama = value;
        }
        //bagian ini digunakan untuk mendapatkan nilai dari variable nama
        get {
                return nama;
        }
}
```

Contoh 4 : Program Konversi Suhu

Konversi.cs (menggunakan property)

```
class Konversi {
    private int celcius;
    public int c {
        set{
            celcius = value;
        }

        get{
            return celcius;
        }
    }

    //Convert to Fahrenheit
    public double Fahrenheit(){
        return (celcius * 1.8 + 32);
    }

    //Convert to Kelvin
    public double Kelvin(){
        return (celcius + 237,15);
    }
}
```

Konversi.cs (menggunakan fungsi set dan get)

```
class Konversi {
    private int celcius;

    //Method Setter
    public void SetNama(int c){
        celcius = c;
    }

    //Method Getter
    public int GetNama(){
        return celcius;
    }

    //Convert to Fahrenheit
    public double Fahrenheit(){
        return (celcius * 1.8 + 32);
    }
```

```
    //Convert to Kelvin
    public double Kelvin(){
        return (celcius + 237,15);
    }
}
```

TampilKonversi.cs

```
class TampilKonversi{
    static void Main(string [] hitung){
        //instance
        Konversi objK = new Konversi();

        //input
        Console.Write("Inputkan suhu (Celcius) : ");
        objK.SetCelcius(Int32.Parse(Console.ReadLine()));

        //output
        Console.WriteLine("Suhu dalam Celcius = {0}", objK.GetCelcius());
        Console.WriteLine("Konversi ke Fahrenheit = {0}", objK.Fahrenheit());
        Console.WriteLine("Konversi ke Kelvin = {0}", objK.Kelvin());
    }
}
```

OVERLOADING FUNGSI

DEFINISI OVERLOADING FUNGSI

Overloading Method adalah terdapat method yang sama pada satu class namun berbeda paramater.

Contoh :

```
class Mobil {
        string warna, merk;

        //contructor default
        public Mobil(){
                warna = "Merah";
                merk = "BMW"
        }

        //overloading Constructor
        public Mobil(string colour, string m){
                warna = colour;
                merk = m;
        }
}
```

IMPLEMENTASI OVERLOADING METHOD

Contoh 5 : Program Menghitung Tegangan

```
class BedaPotensial {
    private int arus, hambatan

    //constructor default
    public BedaPotensial(){
        arus = 0;
        hambatan = 0;
    }

    //constructor parameter (overload constructor)
    public BedaPotensial(int i, int r`){
        arus = i;
        hambatan = r;
    }

    public void Tegangan(){
        int v = arus * hambatan;
        Console.WriteLine("Tegangan = {0}", v);
    }
}
```

Hitung.cs

```
class Hitung{
    static void Main(string [] abc){
        int i, r;
        //input
        Console.Write("arus = ");
        i = Int32.Parse(Console.ReadLine());
        Console.Write("hambatan = ");
        r = Int32.Parse(Console.ReadLine());

        //instance 1 - constructor default
        BedaPotensial bp = new BedaPotensial();

        //output 1
        bp.Tegangan();

        //instance 2 - constructor parameter
        BedaPotensial bp1 = new BedaPotensial(i, r);

        //output 1
        bp1.Tegangan();
    }
}
```

Latihan 1 : Program untuk mencari Faktorial

Buat sebuah program dinamis untuk menentukan Faktorial dari sebuah bilangan/angka.

Contoh :
Input :
Inputkan angka = 5
Output :
5! = 5 x 4 x 3 x 2 x 1
5! = 120

Latihan 2 : Program Gaji Karyawan

Buat sebuah program dinamis untuk menghitung gaji karyawan PT. ABCD Persero.

1. **Class Karyawan memiliki atribut-atribut :**
   NIK, nama, gaji pokok, tunjangan, uang makan, uang transport
2. **Total gaji merupakan penjumlahan seluruh pendapatan yang diperoleh oleh karyawan.**
3. **Buatlah program tersebut dengan method accessor dan mutator.**
4. **Output program berupa data minimal 3 karyawan dengan identitas lengkap, keseluruhan pendapatan dan total gaji yang diperoleh masing-masing karyawan.**

------ Selamat Mengerjakan -----