

LAPORAN PRAKTIKUM

PEMROGRAMAN BERORIENTASI OBJEK

PERTEMUAN KE-6



Disusun Oleh :

NAMA : Raden Isnawan Argi Aryasatya

NIM : 195410257

JURUSAN : Teknik Informatika

JENJANG : S1

KELAS : TI-5

Laboratorium Terpadu
Sekolah Tinggi Manajemen Informatika Komputer
AKAKOM
YOGYAKARTA

2021

PERTEMUAN KE-6

(ENKAPSULASI DAN KOMPOSISI)

TUJUAN

Dapat mengimplementasi konsep enkapsulasi pada aplikasi serta dapat membuat aplikasi yang berelasi komposisi

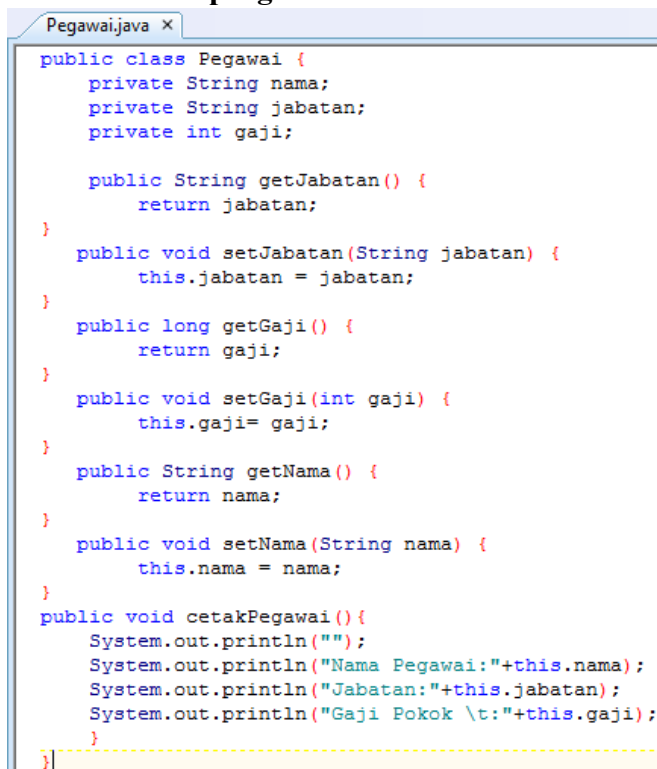
DASAR TEORI

Enkapsulasi Adalah salah satu konsep fundamental dalam object oriented, lainnya adalah pewarisan, polimorfisme dan abstrak. Enkapsulasi adalah suatu cara pengemasan data dan fungsi dalam sebuah kelas yang terlindungi dari akses secara sembarangan dari pihak luar

Komposisi adalah suatu relasi dimana satu kelas bersifat mempunyai(own) kelas yang lain dan apabila pemilik kelas (owner) dihapuskan maka kelas yang lain tidak bisa berfungsi. Contohnya kelas manusia mempunyai kelas kepala, kelas tangan, kelas kaki.

PRAKTIK ENKAPSULASI

1. Ketikkan kode program berikut ini



```
Pegawai.java x
public class Pegawai {
    private String nama;
    private String jabatan;
    private int gaji;

    public String getJabatan() {
        return jabatan;
    }

    public void setJabatan(String jabatan) {
        this.jabatan = jabatan;
    }

    public long getGaji() {
        return gaji;
    }

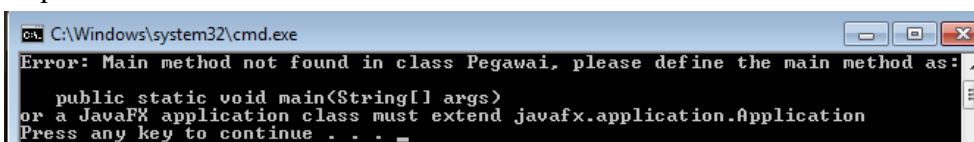
    public void setGaji(int gaji) {
        this.gaji = gaji;
    }

    public String getNama() {
        return nama;
    }

    public void setNama(String nama) {
        this.nama = nama;
    }

    public void cetakPegawai() {
        System.out.println("");
        System.out.println("Nama Pegawai:" + this.nama);
        System.out.println("Jabatan:" + this.jabatan);
        System.out.println("Gaji Pokok \t:" + this.gaji);
    }
}
```

output:



```
C:\Windows\system32\cmd.exe
Error: Main method not found in class Pegawai, please define the main method as:
    public static void main(String[] args)
or a JavaFX application class must extend javafx.application.Application
Press any key to continue . . .
```

penjelasan:

pertama kita membuat class Pegawai yang memiliki modifier public yang fungsinya mendeklarasikan kelas bernama Pegawai yang gunanya untuk mengumpulkan fungsi atau atribut yaitu private String nama; private String jabatan; dan private int gaji;. Lalu kita tulis

baris-baris kode berupa method set dan get. Method setter dan getter adalah dua method yang tugasnya untuk mengambil dan mengisi data ke dalam objek. Dalam encapsulation (pembungkusan), data dibungkus dengan modifier private agar tidak bisa diakses secara langsung dari luar class. Method setter dan getter inilah yang akan membantu kita mengakses data tersebut. Method setter dan getter harus diberikan modifier public, karena method ini akan diakses dari luar class.

Perbedaan method setter dengan getter terletak pada nilai kembalian, parameter, dan isi method-nya.

Contoh penulisan method setter pada script di atas:

```
public void setJabatan(String jabatan) {  
    this.jabatan = jabatan;  
}
```

bisa kita lihat kalau method set tersebut memiliki kembalian void (kosong). Karena tugasnya hanya untuk mengisi data ke dalam atribut, yaitu memasukkan data variabel String jabatan.

Contoh penulisan method getter pada script di atas:

```
public String getJabatan() {  
    return jabatan;  
}
```

method getter memiliki nilai kembalian sesuai dengan tipe data yang akan diambil. Di baris kode di atas getter memiliki nilai kembalian dari variabel getJabatan() { dan setelah itu direturn. Return adalah method yang mengembalikan nilai secara langsung atau sebuah nilai dari variable.

Konsep itu berlaku juga untuk variabel/atribut lainnya yaitu nama dan gaji

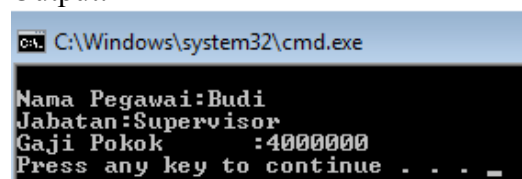
Terakhir kita buat method dengan baris kode **public void cetakPegawai(){** yang fungsinya untuk mencetak dan menampilkan output berupa kalimat dan value/nilai dari variabel-variabel nama, jabatan, dan gaji, yaitu dengan baris-baris kode berikut:

```
System.out.println("");  
System.out.println("Nama Pegawai:"+this.nama);  
System.out.println("Jabatan:"+this.jabatan);  
System.out.println("Gaji Pokok \t:"+this.gaji);
```

2. Atribut nama, jabatan, gaji memiliki hak akses private untuk mempertahankan prinsip enkapsulasi. Mengakses atribut tersebut harus menggunakan method yang hak aksesnya bukan private.

```
32 public class TestEnkapsulasi{  
33     public static void main(String[] args) {  
34         Pegawai dataPeg=new Pegawai();  
35         dataPeg.setNama("Budi");  
36         dataPeg.setJabatan("Supervisor");  
37         dataPeg.setGaji(4000000);  
38         dataPeg.cetakPegawai();  
39     }  
40 }
```

Output:



```
C:\Windows\system32\cmd.exe  
Nama Pegawai:Budi  
Jabatan:Supervisor  
Gaji Pokok :4000000  
Press any key to continue . . . _
```

Setelah kita membuat method setter dan getter, kita bisa mengakses atau menggunakannya seperti method biasa. Yaitu dengan membuat public class bernama TestEnkapsulasi. Lalu deklarasikan **public static void main(String args[]){** untuk mengawali eksekusi. Public, yang berarti metode ini bisa dipanggil dan digunakan didalam Class atau diluar Class. void, yang berarti bahwa metode ini tidak mengirimkan nilai balik. String[], adalah tipe data objek yang menangani serangkaian karakter-karakter yang berjenis array. args, adalah variabel objek.

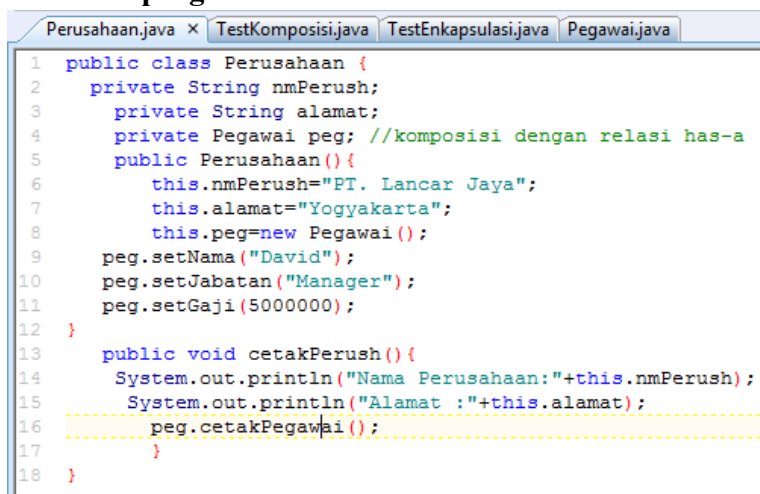
Lalu kita buat objek bernama dataPeg dengan cara menuliskan kode **Pegawai dataPeg=new Pegawai();**. Lalu kita inisialisasikan objek dengan variabel-variabel set yaitu dengan cara:

```
dataPeg.setNama("Budi");  
dataPeg.setJabatan("Supervisor");  
dataPeg.setGaji(4000000);
```

Yang terakhir adalah kita inisialisasi dataPeg dengan method cetakPegawai supaya data yang sudah diisikan di setiap variabel bisa masuk ke kalimat di dalam cetakPegawai yaitu dengan baris kode **dataPeg.cetakPegawai();**.

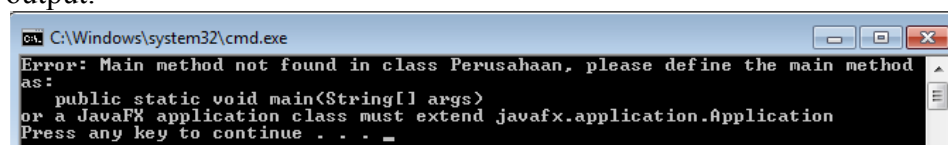
PRAKTIK KOMPOSISI

3. Ketikkan program berikut ini



```
1 public class Perusahaan {  
2     private String nmPerush;  
3     private String alamat;  
4     private Pegawai peg; //komposisi dengan relasi has-a  
5     public Perusahaan(){  
6         this.nmPerush="PT. Lancar Jaya";  
7         this.alamat="Yogyakarta";  
8         this.peg=new Pegawai();  
9         peg.setNama("David");  
10        peg.setJabatan("Manager");  
11        peg.setGaji(5000000);  
12    }  
13    public void cetakPerush(){  
14        System.out.println("Nama Perusahaan:"+this.nmPerush);  
15        System.out.println("Alamat :"+this.alamat);  
16        peg.cetakPegawai();  
17    }  
18 }
```

output:



```
C:\Windows\system32\cmd.exe  
Error: Main method not found in class Perusahaan, please define the main method  
as:  
    public static void main(String[] args)  
or a JavaFX application class must extend javafx.application.Application  
Press any key to continue . . . _
```

penjelasan:

komposisi menggunakan konsep HAS-A (memiliki). Pada komposisi, suatu class akan menggunakan kembali fungsionalitas dengan membuat referensi ke suatu objek class yang ingin digunakan kembali. Contohnya perusahaan memiliki pegawai. setiap objek dari class yang berbeda tidak bisa selalu berhubungan dengan inheritance.

Misalnya pada script program di atas Perusahaan dan class Pegawai, sekarang apakah Perusahaan adalah Pegawai? Tentu saja bukan, jadi konsep inheritance tidak bisa digunakan. Pertanyaannya apakah Perusahaan memiliki Pegawai? Ya, Perusahaan memiliki pegawai. Jadi untuk menghubungkan class semacam ini maka komposisi digunakan.

pertama kita membuat class Perusahaan yang memiliki modifier public (supaya bisa diakses dan mengakses class Pegawai) yang fungsinya mendeklarasikan kelas bernama Perusahaan yang gunanya untuk mengumpulkan fungsi atau atribut yaitu private String nmPerush; private String alamat; dan private Pegawai peg; (ini adalah komposisi nya, diambil dari class Pegawai dan diwakilkan dengan “peg”).

Lalu kita buat konstruktor bernama Perusahaan(). Sudah dijelaskan di modul 4 bahwa konstruktor adalah sebuah method yang namanya sama persis dengan nama class-nya. Constructor sendiri berfungsi untuk memberikan nilai awal pada sebuah class ketika class tersebut dibuat dalam bentuk objek pada class lain. Kemudian di dalam konstruktor Perusahaan() kita inialisasi dan beri value pada variabel/atribut nmPerush, alamat, dan peg yaitu dengan cara menuliskan baris kode:

```
this.nmPerush="PT. Lancar Jaya";
```

```
this.alamat="Yogyakarta";
```

```
this.peg=new Pegawai(); (inisialisasi objek “peg” pada class Pegawai)
```

Setelah itu di bawahnya kita inialisasi dan beri value pada atribut/variabel milik class Pegawai yaitu nama, jabatan, dan gaji dengan menuliskan baris kode:

```
peg.setNama("David");
```

```
peg.setJabatan("Manager");
```

```
peg.setGaji(5000000);
```

Kita buat **public void cetakPerush()** untuk mencetak output, yang di dalamnya ada:

```
System.out.println("Nama Perusahaan:"+this.nmPerush); untuk mencetak nama perusahaan dan inialisasi dengan atribut/variabel nmPerush.
```

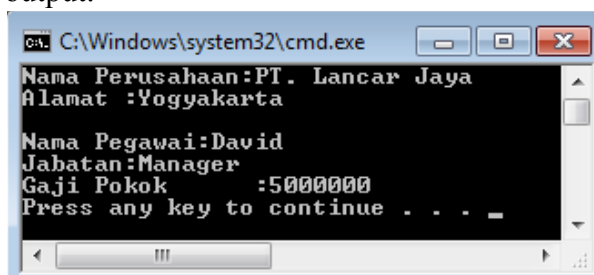
```
System.out.println("Alamat :"+this.alamat); untuk mencetak alamat dan inialisasi dengan atribut/variabel alamat.
```

Lalu yang terakhir **peg.cetakPegawai();** yaitu inialisasi objek peg dengan method cetakPegawai pada class Pegawai yang nantinya berguna untuk mencetak output.

4. Ketikkan program berikut ini

```
public class TestKomposisi{  
    public static void main(String[] args) {  
        Perusahaan kantor1=new Perusahaan();  
        kantor1.cetakPerush();  
    }  
}
```

output:



penjelasan:

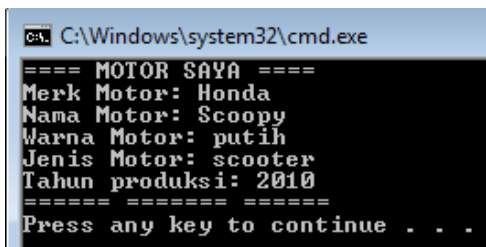
Alasan script nomer 3 tadi tidak mengeluarkan output adalah karena belum ditambahkan script di atas tersebut. Lalu apa fungsi dari baris-baris kode tersebut? Intinya adalah untuk membuat objek. Hal itu dilakukan dengan membuat class yang berfungsi untuk membuat objek yang nanti akan inialisasi dengan cetakPerush(). Pertama, kita buat public class dengan nama **TestKomposisi**. Lalu deklarasikan **public static void main(String args[])**

untuk mengawali eksekusi. Di dalamnya kita buat objek bernama kantor1 dengan kode **Perusahaan kantor1=new Perusahaan();**. Yang terakhir, kita deklarasikan dan inisialisasi objek kantor1 dengan cetakPerush untuk mencetak output dengan **kantor1.cetakPerush();**

LATIHAN

1. Buatlah program Class SepedaMotor yang dilengkapi dengan 5 buah atribut (nama atribut bebas) dan method yang diperlukan, yang merupakan implementasi dari enkapsulasi !

```
MotorScoopy.java x
1  class SepedaMotor {
2      private String merk;
3      private String nama;
4      private String warna;
5      private String jenis;
6      private int tahun;
7
8      public String getMerk() {
9          return merk;
10     }
11     public void setMerk(String merk) {
12         this.merk = merk;
13     }
14     public String getNama() {
15         return nama;
16     }
17     public void setNama(String nama) {
18         this.nama = nama;
19     }
20     public String getWarna() {
21         return warna;
22     }
23     public void setWarna(String warna) {
24         this.warna = warna;
25     }
26     public String getJenis() {
27         return jenis;
28     }
29     public void setJenis(String jenis) {
30         this.jenis = jenis;
31     }
32     public long getTahun() {
33         return tahun;
34     }
35     public void setTahun(int tahun) {
36         this.tahun = tahun;
37     }
38
39     public void cetakMotor(){
40         System.out.println("==== MOTOR SAYA ====");
41         System.out.println("Merk Motor: "+this.merk);
42         System.out.println("Nama Motor: "+this.nama);
43         System.out.println("Warna Motor: "+this.warna);
44         System.out.println("Jenis Motor: "+this.jenis);
45         System.out.println("Tahun produksi: "+this.tahun);
46         System.out.println("===== ");
47     }
48 }
49
50 public class MotorScoopy{
51     public static void main(String[] args) {
52         SepedaMotor Motor1=new SepedaMotor();
53         Motor1.setMerk("Honda");
54         Motor1.setNama("Scoopy");
55         Motor1.setWarna("putih");
56         Motor1.setJenis("scooter");
57         Motor1.setTahun(2010);
58         Motor1.cetakMotor();
59     }
60 }
```



```
C:\Windows\system32\cmd.exe
==== MOTOR SAYA ====
Merk Motor: Honda
Nama Motor: Scoopy
Warna Motor: putih
Jenis Motor: scooter
Tahun produksi: 2010
=====
Press any key to continue . . .
```

Penjelasan:

pertama kita membuat class SepedaMotor yang memiliki modifier public yang fungsinya mendeklarasikan kelas bernama SepedaMotor yang gunanya untuk mengumpulkan fungsi atau atribut yaitu private String merk; private String nama; private String warna; private String jenis; private int tahun;. Lalu kita tulis baris-baris kode berupa method set dan get yang tugasnya untuk mengambil dan mengisi data ke dalam objek.

penulisan method setter pada script di atas:

```
public void setMerk(String merk) {  
    this.merk = merk;
```

method set tersebut memiliki kembalian void (kosong). Karena tugasnya hanya untuk mengisi data ke dalam atribut, yaitu memasukkan data variabel String merk.

penulisan method getter pada script di atas:

```
public String getMerk() {  
    return merk;
```

method getter memiliki nilai kembalian sesuai dengan tipe data yang akan diambil. Di baris kode di atas getter memiliki nilai kembalian dari variabel getMerk() { dan setelah itu direturn. Return adalah method yang mengembalikan nilai secara langsung atau sebuah nilai dari variable. Method ini digunakan untuk mengakses variabel yang telah dienkapsulasi

Konsep itu berlaku juga untuk variabel/atribut nama, warna, jenis, dan tahun

Kita buat method dengan baris kode public void cetakMotor() { yang fungsinya untuk mencetak dan menampilkan output berupa kalimat dan value/nilai dari variabel-variabel merk, nama, warna, jenis, dan tahun dengan baris-baris kode berikut:

```
System.out.println("==== MOTOR SAYA ====");  
System.out.println("Merk Motor: "+this.merk);  
System.out.println("Nama Motor: "+this.nama);  
System.out.println("Warna Motor: "+this.warna);  
System.out.println("Jenis Motor: "+this.jenis);  
System.out.println("Tahun produksi: "+this.tahun);  
System.out.println("===== ");
```

Setelah kita membuat method setter dan getter, kita bisa mengakses atau menggunakannya seperti method biasa. Yaitu dengan membuat public class bernama MotorScoopy. Lalu deklarasikan public static void main(String args[]) { untuk mengawali eksekusi.

Lalu kita buat objek bernama Motor1 dengan cara menuliskan kode SepedaMotor Motor1=new SepedaMotor();. Lalu kita inisialisasikan objek dengan variabel-variabel set yaitu dengan cara menuliskan:

```
Motor1.setMerk("Honda");  
Motor1.setNama("Scoopy");  
Motor1.setWarna("putih");  
Motor1.setJenis("scooter");
```

Motor1.setTahun(2010);

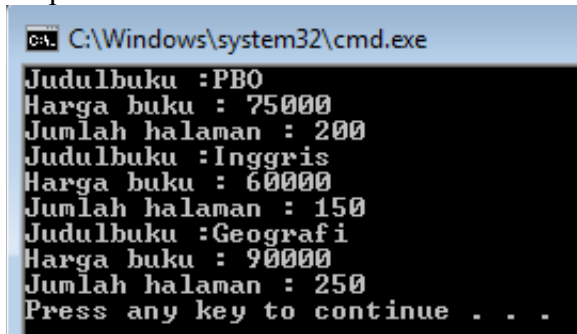
Yang terakhir adalah kita inisialisasi Motor1 dengan method cetakMotor() supaya data yang sudah diisikan di setiap variabel bisa masuk ke kalimat di dalam cetakMotor() dan dicetak yaitu dengan baris kode Motor1.cetakMotor();

TUGAS

1. Modifikasi kelas Buku pada modul 1 supaya program mengimplementasikan enkapsulasi !

```
ObyekBuku.java x
1  class Buku {
2      private String judul;
3      private int harga;
4      private int halaman;
5
6      public String getJudul() {
7          return judul;
8      }
9      public void setJudul(String judul) {
10         this.judul = judul;
11     }
12     public long getHarga() {
13         return harga;
14     }
15     public void setHarga(int harga) {
16         this.harga = harga;
17     }
18     public long getHalaman() {
19         return halaman;
20     }
21     public void setHalaman(int halaman) {
22         this.halaman = halaman;
23     }
24
25     public void tampil()
26     {
27         System.out.println("Judulbuku :"+judul);
28         System.out.println("Harga buku : "+harga);
29         System.out.println("Jumlah halaman : "+halaman);
30     }
31 }
32 public class ObyekBuku
33 {
34     public static void main(String[] args)
35     {
36         Buku buku1=new Buku();
37         buku1.setJudul("PBO");
38         buku1.setHarga(75000);
39         buku1.setHalaman(200);
40         buku1.tampil();
41
42         Buku buku2=new Buku();
43         buku2.setJudul("Inggris");
44         buku2.setHarga(60000);
45         buku2.setHalaman(150);
46         buku2.tampil();
47
48         Buku buku3=new Buku();
49         buku3.setJudul("Geografi");
50         buku3.setHarga(90000);
51         buku3.setHalaman(250);
52         buku3.tampil();
53     }
54 }
```


Output:



```
C:\Windows\system32\cmd.exe
Judulbuku :PBO
Harga buku : 75000
Jumlah halaman : 200
Judulbuku :Inggris
Harga buku : 60000
Jumlah halaman : 150
Judulbuku :Geografi
Harga buku : 90000
Jumlah halaman : 250
Press any key to continue . . .
```

penjelasan:

pertama kita membuat class Buku yang memiliki modifier public yang fungsinya mendeklarasikan kelas bernama Buku yang gunanya untuk mengumpulkan fungsi atau atribut yaitu private String judul; private int harga; private int halaman;. Lalu kita tulis baris-baris kode berupa method set dan get yang tugasnya untuk mengambil dan mengisi data ke dalam objek.

penulisan method setter pada script di atas:

```
public void setMerk(String merk) {  
    this.merk = merk;
```

```
public void setHarga(int harga) {  
    this.harga = harga;
```

```
public void setHalaman(int halaman) {  
    this.halaman = halaman;
```

method set tersebut memiliki kembalian void (kosong). Karena tugasnya hanya untuk mengisi data ke dalam atribut, yaitu memasukkan data variabel String merk, int harga, dan int halaman.

penulisan method getter pada script di atas:

```
public String getMerk() {  
    return merk;
```

```
public long getHarga() {  
    return harga;
```

```
public long getHalaman() {  
    return halaman;
```

method getter memiliki nilai kembalian sesuai dengan tipe data yang akan diambil. Di baris kode di atas getter memiliki nilai kembalian dari variabel getMerk(), getHarga(), getHalaman() dan setelah itu direturn. Return adalah method yang mengembalikan nilai secara langsung atau sebuah nilai dari variable. Method ini digunakan untuk mengakses variabel yang telah dienkapsulasi.

Kita buat method dengan baris kode public void tampil() { yang fungsinya untuk mencetak dan menampilkan output berupa kalimat dan value/nilai dari variabel/atribut merk, harga, halaman yaitu dengan menuliskan baris kode:

```
public void tampil() {  
System.out.println("Judulbuku :"+judul);  
System.out.println("Harga buku : "+harga);  
System.out.println("Jumlah halaman : "+halaman);
```

Setelah itu kita bisa mengakses atau menggunakannya seperti method biasa. Yaitu dengan membuat public class bernama ObyekBuku. Lalu deklarasikan public static void main(String args[]){ untuk mengawali eksekusi.

Kita buat objek bernama buku1 dengan cara menuliskan kode **Buku buku1=new Buku();** lalu kita inisialisasikan objek dengan variabel-variabel set dan method tampil yaitu dengan cara menuliskan:

```
buku1.setJudul("PBO");  
buku1.setHarga(75000);  
buku1.setHalaman(200);  
buku1.tampil();
```

Kita buat objek bernama buku2 dengan cara menuliskan kode **Buku buku2=new Buku();** lalu kita inisialisasikan objek dengan variabel-variabel set dan method tampil yaitu dengan cara menuliskan:

```
Buku buku2=new Buku();  
buku2.setJudul("Inggris");  
buku2.setHarga(60000);  
buku2.setHalaman(150);  
buku2.tampil();
```

Kita buat objek bernama buku3 dengan cara menuliskan kode **Buku buku3=new Buku();** lalu kita inisialisasikan objek dengan variabel-variabel set dan method tampil yaitu dengan cara menuliskan:

```
Buku buku3=new Buku();  
buku3.setJudul("Geografi");  
buku3.setHarga(90000);  
buku3.setHalaman(250);  
buku3.tampil();
```

KESIMPULAN

Di dalam laporan ini saya belajar banyak hal tentang teori, cara, dan teknik mengaplikasikan enkapsulasi dan komposisi dalam program java.

Teknik melakukan enkapsulasi adalah dengan memberikan hak akses private untuk atribut kelas, lalu untuk mengaksesnya dilakukan melalui method yang diberi hak akses public. Jika atribut dideklarasikan dengan hak akses private maka tidak dapat diakses dari luar kelas, maka data tersebut terlindungi. Keuntungan menggunakan enkapsulasi adalah dapat memodifikasi suatu implementasi program tanpa perlu membongkar kode aslinya.

Komposisi adalah mekanisme yang digunakan oleh pemrograman berorientasi objek untuk dapat menggunakan kembali implementasi. Pada komposisi, suatu class akan menggunakan kembali fungsionalitas dengan membuat referensi ke suatu objek class yang ingin digunakan kembali. Contohnya mobil motor memiliki roda, perusahaan memiliki karyawan, dapur memiliki gelas.

Terima Kasih