**MODUL PRAKTIKUM II**
**ENCAPSULATION**

## A. Tujuan

1. Siswa mampu memahami dan menggunakan konsep *Encapsulation* dalam pemrograman java.
2. Siswa mampu memahami konsep i*nformation hiding*.
3. Siswa mampu memahami konsep *interface to acces data*.
4. Siswa mampu memahami konsep *mutator method*.
5. Siswa mampu memahami konsep *assesor method.*
6. Siswa mampu memahami konsep *modifiers.*

## B. Dasar Teori

### *Encapsulation* (Enkapsulasi)

Enkapsulasi adalah suatu cara untuk menyembunyikan informasi detail dari suatu class. Artinya proses membuat paket (memaketkan) data objek bersama dengan metode-metodenya. Proses pembungkusan itu sendiri merupakan cara atau mekanisme untuk melakukan abstraksi. Dalam melakukan pembungkusan kode dan data di dalam java, terdapat tiga tingkat akses yang perlu anda ketahui yaitu private, protected, dan public. Keuntungan menerapkan Encapsulasi yaitu bersifat independen, menghindari efek diluar perencanaan, dan melindungi listing program.

Dua hal yang mendasar dalam enkapsulasi, yakni :

- ***Information Hiding***

Adalah proses yang menyembunyikan informasi dari suatu class sehingga class tersebut tidak dapat diakses dari luar. Yaitu dengan memberikan akses kontrol “private” ketika mendeklarasikan atribut atau method.

- ***Interface to Acces data***

Adalah cara melakukan perubahan terhadap atribut yang di information hiding yaitu dengan membuat suatu interface berupa method untuk menginisialisasi atau merubah nilai dari suatu atribut tersebut.

***Mutator & Assesor Method***

a. ***Accessor Methods***

Untuk mengimplementasikan enkapsulasi, tidak diinginkan sembarang *object* dapat mengakses data kapan saja. Untuk itu, perlu mendeklarasikan atribut dari *class* sebagai *private*. Namun, adakalanya dimana kita menginginkan *object* lain untuk dapat mengakses data *private*. Dalam hal ini digunakan *accessor methods.*

*Accessor Methods* digunakan untuk membaca nilai variable pada *class*, baik berupa *instance* maupun *static*. Sebuah *accessor method* umumnya dimulai dengan penulisan

`get <namaInstanceVariable>`

*Method* ini juga mempunyai sebuah *return value*.

Sebagai contoh, kita ingin menggunakan *accessor method* untuk dapat membaca nama, alamat, dan nilai siswa. Perhatikan salah satu contoh implementasi *accessor method*.

```
public class Student Record{
  private String name;
  :
  :
  public String getName(){
  return name;
  }
}
```

**Keterangan:**

`Public` - menjelaskan bahwa *method* tersebut dapat diakses dari *object* luar class

`String` - tipe data *return value* dari *method* tersebut bertipeString

`getName` - nama dari *method*

`()` - menjelaskan bahwa *method* tidak memiliki parameter apapun

b. ***Mutator Methods***

Bagaimana jika kita menghendaki *object* lain untuk mengubah data yang dapat kita lakukan adalah membuat *method* yang dapat memberi atau mengubah nilai variable

dalam *class*, baik itu berupa *instance* maupun *static*. *Method* semacam ini disebut dengan *mutator methods*. Sebuah *mutator method* umumnya tertulis `set<namaInstanceVariabel>`.

Mari kita perhatikan salah satu dari implementasi *mutator method* :

```
public class StudentRecord
```

```
{
private String name;
:
public void setName( String temp ){
name = temp;}
}
```

**Keterangan:**

`Public` - menjelaskan bahwa *method* ini dapat dipanggil *object* luar class

`Void` - *method* ini tidak menghasilkan *return value*

`setName` - nama dari *method*

`(Stringtemp)` - parameter yang akan digunakan pada *method*

`name=temp` - mengidentifikasi nilai dari *temp* sama dengan *name* dan mengubah data pada *instance variable name*. Perlu diingat bahwa *mutator methods* tidak menghasilkan *return value*. Namun berisi beberapa argumen dari program yang akan digunakan oleh *method*.

Perlu diingat bahwa *mutator methods* tidak menghasilkan *return value*. Namun berisi beberapa argumendari program yang akan digunakan oleh *method*.

***Contructor*** (konstruktor) adalah suatu *method* yang pertama kali dijalankan pada saat pembuatan suatu obyek. Konstruktor mempunyai ciri yaitu:

- ✓ mempunyai nama yang sama dengan nama class,
- ✓ tidak mempunyai return type (seperti void, int, double, dan lain-lain).

Contoh:

Listing Program

```
public class Siswa {
private int nrp;
private String nama;
public Siswa(int n, String m) {
nrp=n;
nama=m;
}
}
```

Suatu *class* dapat mempunyai lebih dari 1 konstruktor dengan syarat daftar parameternya tidak boleh ada yang sama.

Contoh:

Listing Program

```
public class Siswa {
private int nrp;
private String nama;
public Siswa(String m) {
nrp=0;
nama="";
}
public Siswa(int n, String m) {
nrp=n;
```

```
nama=m;
}
}
```

Terdapat 4 macam *access modifiers* di java, yaitu : *public*, *private*, *protected* dan *default*. 3 tipe akses pertama tertulis secara ekplisit pada kode untuk mengindikasikan tipe akses, sedangkan yang keempat yang merupakan tipe default, tidak diperlukan penulisan *keyword* atas tipe akses.

✓ ***Public***

Modifier Public dapat diakses di dalam class itu sendiri, dapat diakses dengan menggunakan metode *extend* dan instan pada paket yang sama, serta dapat diakses dengan metode *extend* maupun instan dalam paket yang berbeda. Artinya hak akses *public* dapat diakses oleh sembarang object.

Data maupun *method* yang bersifat *public* dapat diakses oleh semua bagian di dalam program. Dengan kata lain, data–data maupun *method-method* yang dideklarasikan dengan tingkat akses *public* akan dikenali atau dapat diakses oleh semua kelas yang ada didalam, baik yang merupakan kelas turunan maupun kelas yang tidak memiliki hubungan sama sekali. Untuk mendeklarasikan suatu data atau method dengan tingkat akses *public*, gunakan kata kunci *public*.

Berikut contoh program sederhana :

Listing Program

```
class atas
{
public int a;
protected int b;
private int c;
}
class bawah{
public static void main(String[]args){
atas objek = new atas();
objek.a=2;
objek.b=3;
System.out.println("nilai a: "+objek.a);
System.out.println("nilai b: "+objek.b);
}
}
```

Penjelasan: program di atas terdiri dari dua kelas yaitu kelas sekunder yang berisi variabel a, b dan c dengan tingkat akses yang berbeda, dan kelas primer yang berisi objek untuk melakukan *instance* pada kelas turunan, objek pada kelas primer hanya dapat mengisi nilai pada variabel a dan b karena kedua variabel tersebut memiliki

tingkat akses *public* dan *protected*, karena variabel c memiliki tingkat akses *private* maka obyek pada kelas primer tidak bisa mengisi variabel tersebut.

✓ ***Protected***

Suatu data maupun *method* yang dideklarasikan dengan tingkat akses *protected* dapat diakses oleh kelas yang memilkinya dan juga oleh kelas–kelas yang masih memiliki oleh hubungan turunan. Sebagai contoh, apabila data x dalam kelas A dideklarasikan sebagai *protected*, maka kelas B (yang merupakan turunan dari kelas A) diizinkan untuk mengakses data x. Namun apabila terdapat kelas lain, misalnya C (yang bukan merupakan turunan dari kelas A maupun B), tetap tidak dapat mengakses data – data yang dideklarasikan dengan tingkat akses *protected*. Untuk mendeklarasikan suatu data atau *method* dengan tingkat akses *protected*, gunakan kata kunci *ptotected*.

Listing Program

```
public class motor
{
    protected String jenismotor;
    protected String address;
    public motor()
        program turunan:
        program honda.java
        public class honda extends motor
        {
            protected String jenishonda;
            protected String kecepatanhonda;
            public honda()
            {
```

dari contoh program protected yang dapat mengakses hanya kelas motor dan kelas turunannya, yaitu Honda.

✓ ***Private***

Dengan mendeklarasikan data dan method menggunakan tingkat akses private, maka data dan *method* tersebut hanya dapat diakses oleh kelas yang memilikinya saja. Ini berarti data dan *method* tersebut tidak boleh diakses atau digunakan oleh kelas-kelas lain yang terdapat didalam program. Untuk mendeklarasikan suatu data atau *method* dengan tingkat akses *private*, gunakan kata kunci *private*.

Listing Program

```
public class Siswa
{
```

```
//akses dasar terhadap variabel
private String nama;
//akses dasar terhadap metode
private String getNama(){
return name;
}
}
```

Pada contoh diatas, variabel nama dan *method* getNama() hanya dapat diakses oleh *method internal class* tersebut.

✓ ***Default***

Untuk hak akses *default* ini, sebenarnya hanya ditujukan untuk *class* yang ada dalam satu paket, atau istilahnya hak akses yang berlaku untuk satu folder saja (tidak berlaku untuk *class* yang tidak satu folder/package).

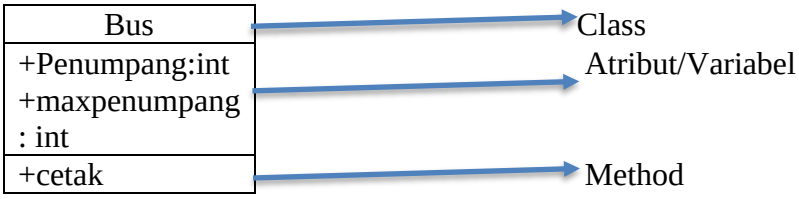
Listing Program

```
public class Siswa{
//akses dasar terhadap variabel
String nama;

//akses dasar terhadap method
String getName(){
return nama;
}
}
```

Pada contoh diatas, variabel nama dan *method* getNama() hanya dapat diakses oleh *method internal class* tersebut.

## C. Latihan

1. Latihan 1




Tanda + menandakan public

a. Buatlah program new project pada netbeans
b. Buatlah class dengan nama Bus dan ketikkan program berikut :

```
public class Bus {
    public int penumpang,maxpenumpang;


    public void cetak(){
        System.out.println("Penumpang sekarang = "+penumpang);
```

```
        System.out.println("penumpang seharusnya adalah ="+maxpen
umpang);
    }
}
```

c. Buatlah class dengan nama UjiBus dan ketikkan program berikut:

```
public class UjiBus {
    public static void main(String[] args) {
        // TODO code application logic here
        Bus busMini=new Bus();
        busMini.penumpang=5;
        busMini.maxpenumpang=5;
        busMini.cetak();

        busMini.penumpang=busMini.penumpang+5;
        busMini.cetak();

        busMini.penumpang=busMini.penumpang-2;
        busMini.cetak();


        busMini.penumpang=busMini.penumpang+8;
        busMini.cetak();


    }
```

c. Output

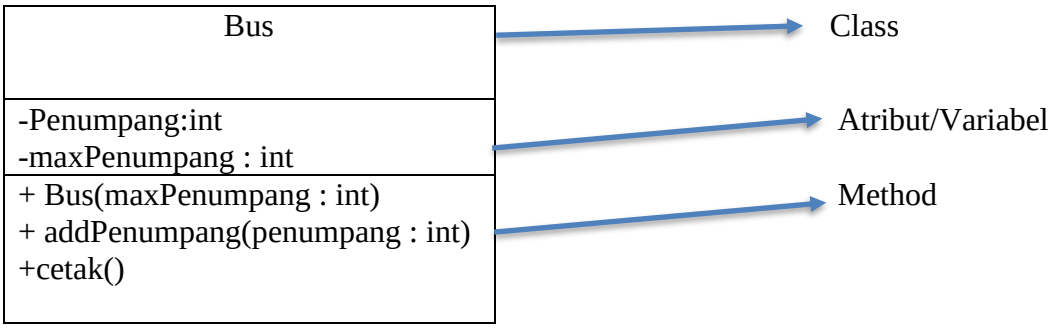
```
Penumpang sekarang = 5
Penumpang seharusnya adalah = 5
Penumpang sekarang = 10
Penumpang seharusnya adalah = 5
Penumpang sekarang = 8
Penumpang seharusnya adalah = 5
Penumpang sekarang = 16
Penumpang seharusnya adalah = 5
```

d. Perhatikan jumlah penumpang Bus terakhir dengan jumlah penumpang maksimum. Mengapa bisa demikian?

Jumlah penumpang berubah sementara max penumpang tetap 5

2. Latihan 2

a. Atribut Penumpang dan maxPenumpang akan diubah menjadi private




b. Ubah pada class Bus.java seperti ini:

```
public class Bus {
    private int penumpang,maxpenumpang;

    // konstruktor
    public Bus(int maxpenumpang){
        this.maxpenumpang=maxpenumpang;
        penumpang = 0;
    }

    //method mutator
    public void pluspenumpang(int penumpang){
        int temp;
        temp=this.penumpang+penumpang;
        if (temp>=maxpenumpang){
            System.out.println("Overload penumpan
g");
        }
        else {
            this.penumpang=temp;
        }

    }
    public void cetak(){
        System.out.println("Penumpang sekarang =
"+penumpang);
        System.out.println("penumpang seharusnya
adalah ="+maxpenumpang);
    }
}
```

b. Jalankan file UjiBus.java. Keluaran program adalah

```
run:
java.lang.ExceptionInInitializerError
Caused by: java.lang.RuntimeException: Uncompilable code - reached end of file while parsing
        at moduledua.UjiBus.<clinit>(UjiBus.java:1)
```

c. Berikan penjelasan point b!

Kode tidak bisa dikompile

d. Ubah class UjiBus menjadi berikut:

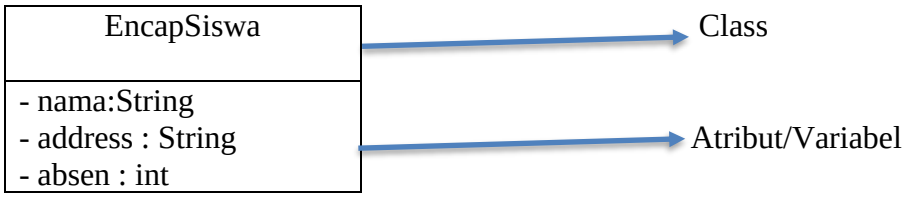
```
public class UjiBus {

    public static void main(String[] args) {
        // TODO code application logic here
        Bus busMini=new Bus(10);
        busMini.cetak();

        busMini.pluspenumpang(3);
        busMini.cetak();

        busMini.pluspenumpang(1);
        busMini.cetak();

        busMini.pluspenumpang(1);
        busMini.cetak();


    }
}
```

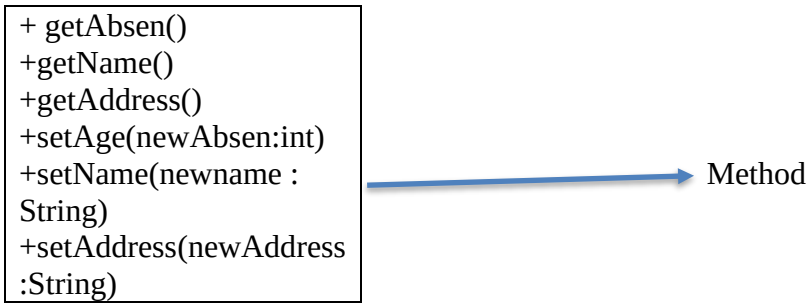
e. Jalankan kembali class UjiBus, Output dari program adalah:

```
run:
Penumpang sekarang = 0
penumpang seharusnya adalah =10
Penumpang sekarang = 3
penumpang seharusnya adalah =10
Penumpang sekarang = 4
penumpang seharusnya adalah =10
Penumpang sekarang = 5
penumpang seharusnya adalah =10
```

3. Latihan 3

Kita akan membuat project dengan menerapkan information hiding (private) dan mutator method serta accessor method






a) Buatlah program new project pada netbeans

b) Buatlah class dengan nama EncapSiswa dan ketikkan program berikut :

```
public class encapsiswa {
    private String name;
    private String address;
    private int age;

    public int getAge(){
        return age;
    }
    public String getName(){
        return name;
    }
    public String getAddress(){
        return address;
    }
    public void setAge (int newAge){
        age=newAge;
    }
    public void setName (String newName){
        name=newName;
    }
    public void setAddress (String newAddress){
        address=newAddress;
    }

}
```

c) Buatlah class dengan nama TestSiswa dan ketikkan program berikut :

```
public class TestSiswa {


    public static void main(String[] args) {
        // TODO code application logic here
    encapsiswa siswa = new encapsiswa();
    siswa.setName("agus");
    siswa.setAge(20);
```

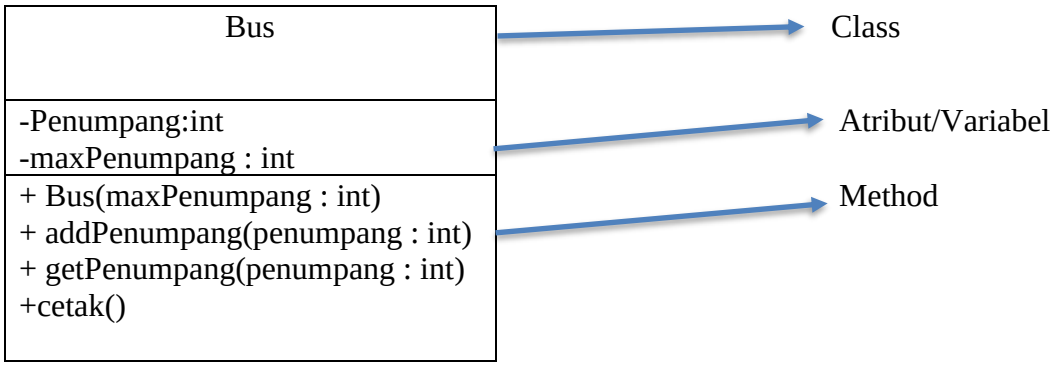
```
    siswa.setAddress("Malang");

        System.out.println("nama: "+siswa.getName()
        + " alamat "+siswa.getAddress()+ " berumur "+siswa.getAge(
)
        + " tahun");
    }

}
```

b. Screenshoot hasil Output

```
nama: agus alamat Malang berumur 20 tahun
BUILD SUCCESSFUL (total time: 0 seconds)
```

4. Latihan 4

a. Tambahkan method getPenumpang pada class Mobil. Tambahkan aturan untuk mengakses data penumpang baru ke dalam method getPenumpang. Aturan yang ditambahkan memuat kode akses(password). Jika password benar, maka data penumpang yang baru ditambahkan dan ditampilkan, jika password salah, maka ada peringatan bahwa password salah.




Sehingga akan dihasilkan program seperti ini pada class Bus:

```
public class Bus {
    private int penumpang,maxpenumpang;

    // konstruktor
    public Bus(int maxpenumpang){
        this.maxpenumpang=maxpenumpang;
        penumpang = 0;
    }
```

```
    //method mutator
    public void pluspenumpang(int penumpang){
        int temp;
        temp=this.penumpang+penumpang;
        if (temp>=maxpenumpang){
            System.out.println("Overload penumpang");
        }
        else {
            this.penumpang=temp;
        }

    }
    public void getPassword(int password){
        if (password==90){
            System.out.println("password benar");
        }
        else{
            System.out.println("pass salah");
        }
    }
    public void cetak(){
        System.out.println("Penumpang sekarang = "+penumpang);
        System.out.println("penumpang seharusnya adalah ="+maxpenumpang);
    }
}
```

dan pada class ujiBus seperti ini:

```
public class UjiBus {
    public static void main(String[] args) {
        // TODO code application logic here
        Bus busMini=new Bus(10);
        busMini.getPassword(40);
        busMini.getPassword(90);
        busMini.cetak();

        busMini.pluspenumpang(3);
        busMini.cetak();

        busMini.pluspenumpang(1);
        busMini.cetak();

        busMini.pluspenumpang(1);
```

```
        busMini.cetak();


    }
}
```

b. Screenshot Output Program tersebut adalah

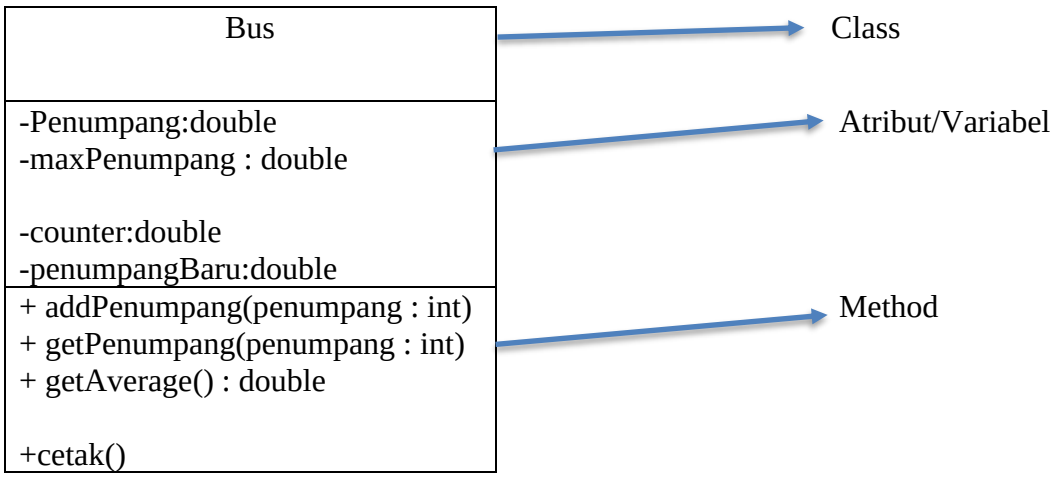
```
pass salah
password benar
Penumpang sekarang = 0
penumpang seharusnya adalah =10
Penumpang sekarang = 3
penumpang seharusnya adalah =10
Penumpang sekarang = 4
penumpang seharusnya adalah =10
Penumpang sekarang = 5
penumpang seharusnya adalah =10
```

c. Penjelasan Program tersebut adalah

Program menggunakan class Bus kemudian mencoba membuka password, setelah itu program mencetak jumlah penumpang sekarang kemudian menambah/kurangkannya

## D. Tugas Praktikum

1. Buat program dengan ketentuan berikut



Dari hasil program latihan 4, tambahkan method getAverage() untuk menghitung rata-rata berat penumpang yang ditambahkan !

2. Buatlah sebuah class “Bola” dengan property *jari-jari* dan method *setJarijari(), showDiameter(), showLuasPermukaan(),* dan *showVolume().* Gunakan double untuk presisi variable *jari-jari* dan gunakan library math class untuk menggunakan constanta Math.PI.
Selanjutnya buatlah file testnya dengan urutan langkah, membuat variable *Jarijari*, menciptakan objek Bola, memanggil method *showDiameter(), showLuasPermukaan(),* dan *showVolume().* Selanjutnya memanipulasi objek yang telah dibuat dengan memanggil method setJarijari() dengan menggunakan variable *Jarijari* yang telah dibuat. Tampilkan lagi hasilnya dengan menggunakan method *showDiameter(), showLuasPermukaan(),* dan *showVolume().* (Jangan lupa untuk menggunakan konsep enkapsulasi)

**Puncak dari Keberhasilan Bukan Semata Karena Nilai yang Bagus,**
**Melainkan Perwujudan Pengetahuan dengan Budi Pekerti yang Luhur**

**--Anak Telkom Keren--**