



Pemrograman Berorientasi Objek

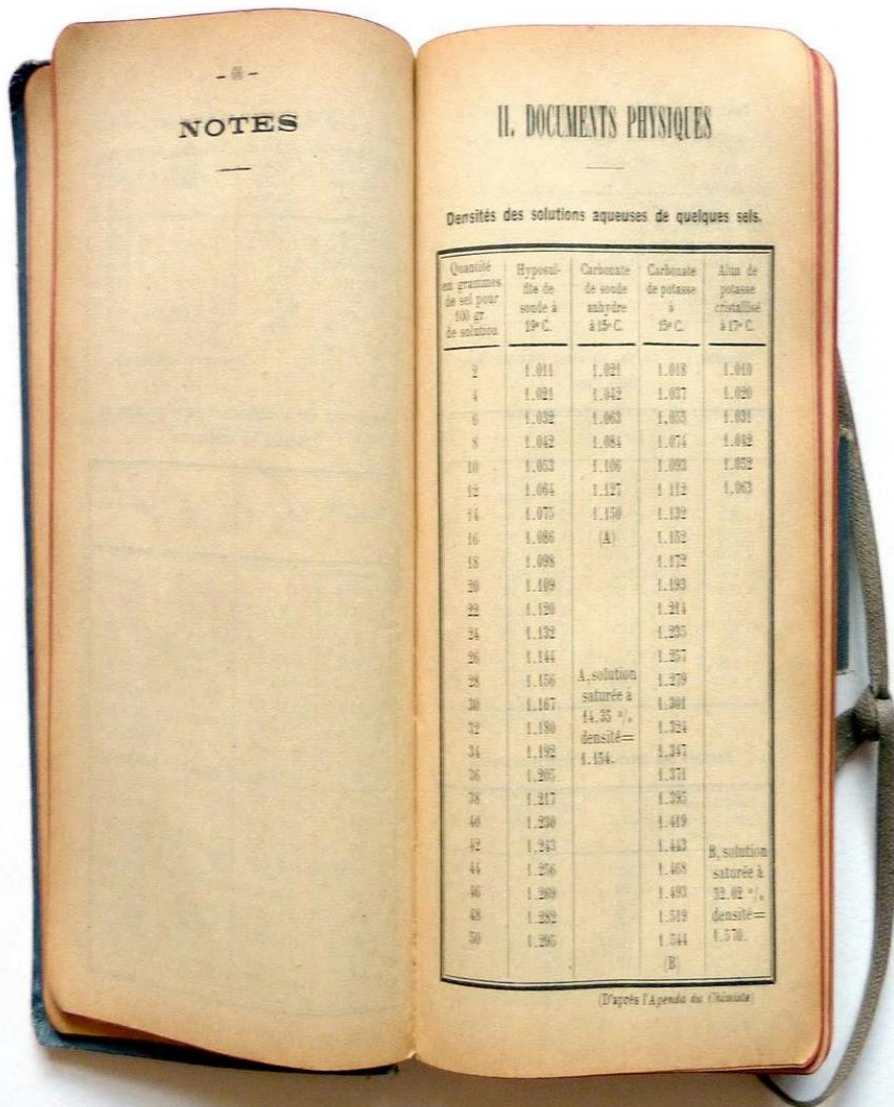


Pertemuan 4

Konsep Dasar Object & Class (bag 2)

Pemateri : Chrismikha Hardyanto S.Kom., M.Kom.

KONTEN PERKULIAHAN



- Package pada JAVA
- Overloading Method
- Object sebagai Parameter & Argumen pada Method
- Konsep Enkapsulasi (access modifier)
- Membuat Getter & Setter
- Memanfaatkan Class Scanner

Minggu lalu kita sudah belajar tentang **class** dan **object** pada JAVA, selanjutnya ada beberapa **konsep** yang dapat Kita manfaatkan didalam **Pemrograman Object**

Package

- ❑ Saat kita membuat program pada JAVA, bisa dipastikan kita akan **banyak sekali membuat class**.
- ❑ Jika class yang dibuat terlalu banyak, terkadang akan **menyulitkan programmer** untuk mencari atau mengelompokkan jenis-jenis class untuk kebutuhan program.
- ❑ JAVA memiliki sebuah fitur yaitu **package**, Package mirip seperti **membuat folder/direktori** yang bisa digunakan untuk menyimpan seluruh class-class didalam program.
- ❑ Sama seperti folder/direktori, package juga dapat dibuat **nested(bersarang)**. Kita bisa menggunakan **tanda titik (.)** untuk membuat nested package.

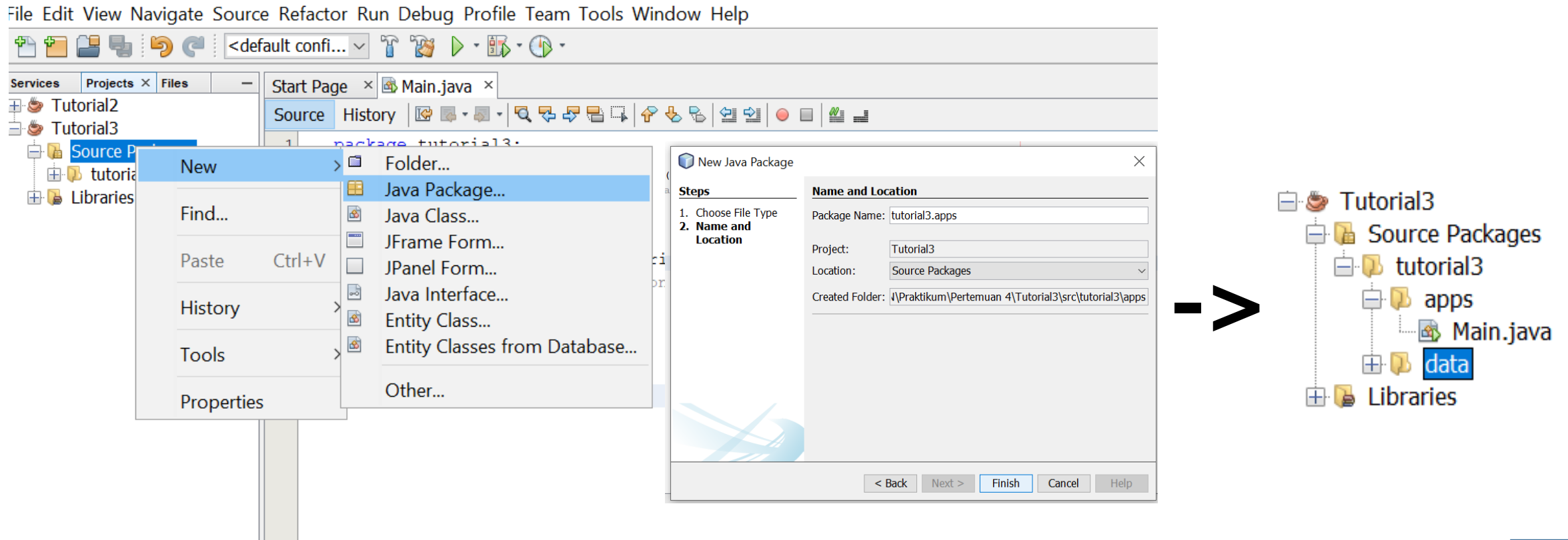


{OOP}

Membuat Package

- ❑ Untuk membuat package pada Netbeans, caranya cukup mudah. **Langkah – langkahnya** adalah sebagai berikut :

Sorot project -> klik kanan -> new Java Package -> Masukan nama package -> Finish



*Ketika memberikan nama package pada program JAVA ada beberapa **aturan/kebiasaan** yang sering dilakukan oleh **programmer JAVA**. Silakan anda cari tahu **bagaimana menulis nama package yang baik di JAVA**

Medefinisikan Package Pada Class

- ❑ Ketika kita **membuat/menyimpan class** didalam suatu package, maka **dias file JAVA** nya kita **wajib menyebutkan** nama packagenya.
- ❑ Namun jika kita membuat file class pada **netbeans**, maka pendefinisian package akan **digenerate otomatis**.

```
package tutorial3.apps;
```

```
public class Produk*{lokasi file Produk.Java disimpan  
    String nama;  
    int harga;  
    int jumlah;  
  
    Produk(String nama, int harga){  
        this.nama = nama;  
        this.harga = harga;  
    }  
  
}
```

*Jika pendefinisian package pada class **salah** (lokasi tidak sesuai), maka program akan error

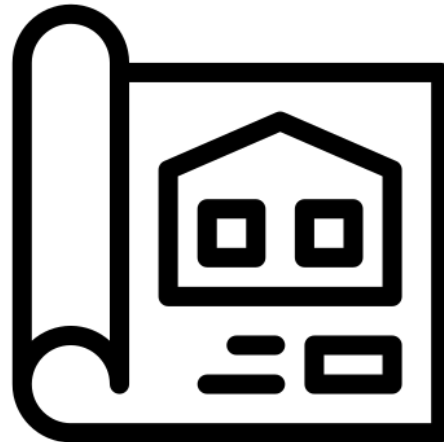
{OOP}

Mengenal konsep

OVERLOADING METHOD

Overloading Method

- ❑ Kemampuan untuk membuat **method** dengan nama yang sama **lebih dari sekali** (2 atau lebih method) didalam sebuah class
- ❑ Namun ada ketentuan yang harus dipenuhi, yaitu **parameter method-nya harus berbeda** baik dari **jumlah parameternya** atau **tipe data parameternya**.
- ❑ Jika ada method yang **data parameternya sama**, maka program JAVA tersebut akan **error**



{OOP}

Ilustrasi Overloading Pada JAVA

```
//Contoh OVERLOADING METHOD
void sayHello(){
    System.out.println("Hallo");
}

void sayHello(String namaDepan){
    System.out.println("Hallo " +namaDepan);
}

void sayHello(String namaDepan, String namaBelakang){
    System.out.println("Hallo " +namaDepan +" "+namaBelakang);
}
```

*Program JAVA tidak akan **error** jika Kita membuat method dengan nama yang sama, asalkan **isi parameternya** berbeda. Itulah kemampuan **Overloading Method**

Contoh Overloading Pada JAVA

```
public class BangunDatar {  
  
    //Contoh 1 Overloading, Method untuk Mencetak Jenis Bangun Datar  
    void tampilBangunDatar(int sisi){  
        System.out.println("Bangun Datar ini adalah Persegi\n"  
            + "dengan sisi : "+sisi +" yang bertipe data Integer");  
    }  
  
    void tampilBangunDatar(double sisi){  
        System.out.println("Bangun Datar ini adalah Persegi\n"  
            + "dengan sisi : " +sisi +" yang bertipe data Double");  
    }  
  
    void tampilBangunDatar(int panjang, int lebar){  
        System.out.println("Bangun Datar ini adalah Persegi Panjang\n"  
            + "dengan panjang : " +panjang+" & Lebar : " +lebar+" "  
            + "\nyang bertipe data Integer");  
    }  
  
    void tampilBangunDatar(double panjang, double lebar){  
        System.out.println("Bangun Datar ini adalah Persegi Panjang\n"  
            + "dengan panjang : " +panjang+" & Lebar : " +lebar+" "  
            + "\nyang bertipe data Double");  
    }  
  
}
```

Untuk mencoba konsep overloading, Mari kita buat **sebuah kelas** dengan nama **BangunDatar** lalu isi classnya seperti contoh berikut.

Contoh Overloading Pada JAVA

Untuk melihat bagaimana hasilnya, mari kita panggil tiap method pada kelas BangunDatar di kelas Main.
(instansiasi object-nya terlebih dahulu)

[Kelas main]

```
public static void main(String[] args) {  
  
    //Intansiasi sebuah object dari Kelas BangunDatar  
    BangunDatar persegi= new BangunDatar();  
  
    persegi.tampilBangunDatar(10);  
    System.out.println("");  
  
    persegi.tampilBangunDatar(7.5);  
    System.out.println("");  
  
    persegi.tampilBangunDatar(10, 5);  
    System.out.println("");  
  
    persegi.tampilBangunDatar(5.2, 6.7);  
    System.out.println("");  
}
```

run:

Bangun Datar ini adalah Persegi
dengan sisi : 10 yang bertipe data Integer

Bangun Datar ini adalah Persegi
dengan sisi : 7.5 yang bertipe data Double

Bangun Datar ini adalah Persegi Panjang
dengan panjang : 10 & Lebar : 5
yang bertipe data Integer

Bangun Datar ini adalah Persegi Panjang
dengan panjang : 5.2 & Lebar : 6.7
yang bertipe data Double

BUILD SUCCESSFUL (total time: 0 seconds)

*Biarapun nama methodnya sama, namun **method yang diakses oleh JVM tidak akan salah**

Contoh Overloading Pada JAVA (2)

```
public class BangunDatar {

    //Contoh 1 Overloading, Method untuk Mencetak Jenis Bangun Datar
    void tampilBangunDatar(int sisi){...5 lines }
    void tampilBangunDatar(double sisi){...5 lines }
    void tampilBangunDatar(int panjang, int lebar){...6 lines }
    void tampilBangunDatar(double panjang, double lebar){...6 lines }

    //Contoh 2 Overloading, Method untuk menghitung luas pada kelas BangunDatar
    int hitungLuas(int sisi){
        int luasPersegi = sisi * sisi;
        return luasPersegi;
    }

    double hitungLuas(double sisi){
        double luasPersegi = sisi * sisi;
        return luasPersegi;
    }

    int hitungLuas(int panjang, int lebar){
        int luasPersegiPanjang = panjang * lebar;
        return luasPersegiPanjang;
    }

    double hitungLuas(double panjang, double lebar){
        double luasPersegiPanjang = panjang * lebar;
        return luasPersegiPanjang;
    }

}
```

Contoh lainnya dari penggunaan overloading method didalam program, tambahkan **method hitungLuas() pada class BangunDatar**

Contoh Overloading Pada JAVA (2)

Untuk melihat bagaimana hasilnya, mari kita panggil tiap method `hitungLuas()` pada kelas `BangunDatar` di kelas `Main`.

[Kelas main]

```
public static void main(String[] args) {  
  
    //Intansiasi sebuah object dari Kelas BangunDatar  
    BangunDatar persegi= new BangunDatar();  
  
    persegi.tampilBangunDatar(10);  
    System.out.println("Luas Bangun Datar = " +persegi.hitungLuas(10));  
    System.out.println("");  
  
    persegi.tampilBangunDatar(7.5);  
    System.out.println("Luas Bangun Datar = " +persegi.hitungLuas(7.5));  
    System.out.println("");  
  
    persegi.tampilBangunDatar(10, 5);  
    System.out.println("Luas Bangun Datar = " +persegi.hitungLuas(10,5));  
    System.out.println("");  
  
    persegi.tampilBangunDatar(5.2, 6.7);  
    System.out.println("Luas Bangun Datar = " +persegi.hitungLuas(5.2,6.7));  
    System.out.println("");  
}
```

run:

Bangun Datar ini adalah Persegi
dengan sisi : 10 yang bertipe data Integer
Luas Bangun Datar = 100

Bangun Datar ini adalah Persegi
dengan sisi : 7.5 yang bertipe data Double
Luas Bangun Datar = 56.25

Bangun Datar ini adalah Persegi Panjang
dengan panjang : 10 & Lebar : 5
yang bertipe data Integer
Luas Bangun Datar = 50

Bangun Datar ini adalah Persegi Panjang
dengan panjang : 5.2 & Lebar : 6.7
yang bertipe data Double
Luas Bangun Datar = 34.84

BUILD SUCCESSFUL (total time: 0 seconds)

Overloading Konstruktor

- ❑ Sama seperti di method, konstruktor pun bisa menerapkan **Overloading**.
- ❑ Kita bisa membuat **Konstruktor lebih dari satu**, dengan syarat **tipe data parameternya berbeda** atau **jumlah parameternya berbeda** (sama seperti syarat pada overloading method)
- ❑ Overloading Konstruktor dapat membantu pembuatan object didalam program menjadi **lebih fleksibel** sesuai kebutuhan.

{OOP}

Ilustrasi Overloading Konstruktor

```
public class Manusia {  
  
    String nama;  
    String alamat;  
    int umur;  
  
    //Deklarasi Konstruktor dari Class Manusia  
    Manusia() {  
  
    }  
  
    Manusia(String nama) {  
        this.nama = nama;  
    }  
  
    Manusia(String nama, String alamat, int umur) {  
        this.nama = nama;  
        this.alamat = alamat;  
        this.umur = umur;  
    }  
}
```

[Kelas main]

```
public static void main(String[] args) {  
  
    //Instansiasi Object dari Class manusia  
    //Dengan menggunakan Konstruktor yang berbeda  
  
    Manusia manusia1 = new Manusia();  
    Manusia manusia2 = new Manusia ("Chrismikha");  
    Manusia manusia3 = new Manusia ("Eko", "Cibiru", 37);  
}
```

*Object yang diinstansiasi nantinya akan menjalankan constructor **sesuai dengan argumen** yang diberikan didalamnya

Contoh Overloading Konstruktor

```
public class BangunDatar {
    //Deklarasi Konstruktor dari Class BangunDatar
    BangunDatar() {

    }

    BangunDatar(int sisi){
        tampilBangunDatar(sisi);
    }

    BangunDatar(double sisi){
        tampilBangunDatar(sisi);
    }

    BangunDatar(int panjang, int lebar){
        tampilBangunDatar(panjang, lebar);
    }

    //Contoh 1 Overloading, Method untuk Mencetak Jenis Bangun Datar
    void tampilBangunDatar(int sisi){...4 lines }
    void tampilBangunDatar(double sisi){...4 lines }
    void tampilBangunDatar(int panjang, int lebar){...5 lines }
    void tampilBangunDatar(double panjang, double lebar){...5 lines }

    //Contoh 2 Overloading, Method untuk menghitung luas pada kelas BangunDatar
    int hitungLuas(int sisi){...4 lines }
    double hitungLuas(double sisi){...4 lines }
    int hitungLuas(int panjang, int lebar){...4 lines }
    double hitungLuas(double panjang, double lebar){...4 lines }
}
```

Implementasi overloading constructor didalam program, Setiap constructor akan menjalankan method tampilBangunDatar() sesuai dengan **tipe data** atau **jumlah parameter** yang dipassing

Contoh Overloading Konstruktor

Untuk melihat bagaimana hasilnya, mari kita buat **beberapa object** dari kelas BangunDatar di kelas Main dengan **mengisi argumen yang berbeda** setiap object diinstansiasi.

[Kelas main]

```
System.out.println("Contoh Overloading Konstruktor");
System.out.println("");

//Contoh Pembuatan object dengan Konstruktor Berparameter
BangunDatar persegi2 = new BangunDatar(20);
System.out.println("");

BangunDatar persegi3 = new BangunDatar(12.5);
System.out.println("");

BangunDatar persegi4 = new BangunDatar(20, 5);
```

Contoh Overloading Konstruktor

Bangun Datar ini adalah Persegi
dengan sisi : 20 yang bertipe data Integer

Bangun Datar ini adalah Persegi
dengan sisi : 12.5 yang bertipe data Double

Bangun Datar ini adalah Persegi Panjang
dengan panjang : 20 & Lebar : 5
yang bertipe data Integer
BUILD SUCCESSFUL (total time: 0 seconds)

*Object yang diinstansiasi mampu **menjalankan konstruktor yang sesuai** dengan nilai argumen yang diinputkan pada pembuatan object

Selanjutnya kita lihat bagaimana menggunakan
OBJECT sebagai **Parameter & Argumen**

Objek Sebagai Parameter



[sumber : Modul PBO – Brigida Arie Minartiningtyas, M.Kom]

Dalam Java, objek juga dapat berperan sebagai parameter dari sebuah method

Object Sebagai Parameter

- ❑ Didalam JAVA, ketika kita membuat sebuah method maka kita juga dapat **mengisi nilai parameter** didalam method tersebut dengan suatu **object**.
- ❑ Object sebagai parameter nantinya dapat menerima **nilai referensi (References)** dari object yang dipassing pada **argumen** ketika **method tersebut dipanggil**
- ❑ **Method** yang menerima parameter object nantinya akan dapat menggunakan **seluruh isi** dari object saat ini (yang dipassing). Sangat bermanfaat untuk **komunikasi data** antar class

{OOP}

Object Sebagai Parameter

- ❑ Bentuk umum pembuatan method dengan object sebagai parameternya adalah sebagai berikut :

```
<type method> namaMethod (Kelas namaObjek) {  
    //Block Program didalam method  
}
```

- ❑ Contoh implementasinya Pada JAVA :

```
Kotak (Kotak kotak) {  
    this.panjang = kotak.panjang;  
    this.lebar = kotak.lebar;  
}
```

{OOP}

Melewatkan Argumen di dalam Java

[sumber : Modul PBO – Brigida Arie Minartiningtyas, M.Kom]



Di dalam java kita tidak dapat secara eksplisit melewati parameter berdasarkan nilai atau referensinya seperti yang dapat kita lakukan pada Pascal ataupun C++

Pass by Value : parameter berupa tipe data sederhana (int, cha, boolean, dll)
Pass by References : parameter berupa objek

Object Sebagai Argumen

- ❑ Apabila kita membuat sebuah method dengan menggunakan **object sebagai parameter**, maka ketika ingin **memanggil method** tersebut kita wajib membuat argumen berupa **object**
- ❑ Object pada argumen berisi **nilai references** dari object saat ini yang digunakan/dipanggil didalam method
- ❑ Yang perlu diingat bahwa Objek yang digunakan sebagai argumen **dapat dirubah/dimanipulasi** nilai atributnya didalam method yang menerimanya. (apabila ada kode program untuk mengubah isi atribut)

{OOP}

Contoh Object sebagai parameter dan Argumen

Untuk melihat bagaimana membuat dan memanggil method dengan parameter suatu object, perhatikan penggalan program JAVA berikut :

```
public class Kotak {

    int panjang;
    int lebar;

    Kotak() {

    }

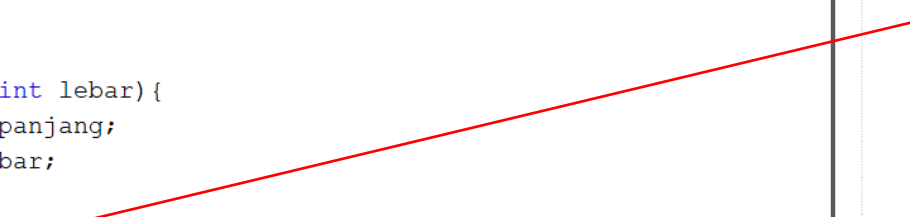
    Kotak(int panjang, int lebar) {
        this.panjang = panjang;
        this.lebar = lebar;
    }

    Kotak(Kotak kotak) {
        this.panjang = kotak.panjang;
        this.lebar = kotak.lebar;
    }

    void tampilKeteranganKotak() {

        System.out.println("Panjang Kotak ini adalah : " +this.panjang);
        System.out.println("Lebar Kotak ini adalah : " +this.lebar);
    }

}
```



[Kelas main]

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

    //Instansiasi Object kotak1, kotak2, kotak3
    Kotak kotak1 = new Kotak(5,4);
    Kotak kotak2 = new Kotak(10,8);
    Kotak kotak3 = new Kotak(kotak1);

    //Menampilkan nilai panjang & lebar setiap object Kotak
    System.out.println("Tampil Nilai Object Kotak ke-1");
    System.out.println("=====");
    kotak1.tampilKeteranganKotak();

    System.out.println("Tampil Nilai Object Kotak ke-2");
    System.out.println("=====");
    kotak2.tampilKeteranganKotak();

    System.out.println("Tampil Nilai Object Kotak ke-3");
    System.out.println("=====");
    kotak3.tampilKeteranganKotak();

}
```

*Object **kotak3** akan memiliki **nilai panjang & lebar yang sama** dengan milik object **kotak1**

Contoh Output Program

```
run:
Tampil Nilai Object Kotak ke-1
=====
Panjang Kotak ini adalah : 5
Lebar Kotak ini adalah   : 4
Tampil Nilai Object Kotak ke-2
=====
Panjang Kotak ini adalah : 10
Lebar Kotak ini adalah   : 8
Tampil Nilai Object Kotak ke-3
=====
Panjang Kotak ini adalah : 5
Lebar Kotak ini adalah   : 4
BUILD SUCCESSFUL (total time: 0 seconds)
```

{OOP}

Selanjutnya mari kita bahas tentang Konsep Dasar
ENKAPSULASI didalam **PBO**

Konsep Enkapsulasi

- ❑ **Enkapsulasi** adalah konsep **pembungkusan informasi** yang dimiliki oleh suatu **data**
- ❑ Konsep ini bertujuan agar informasi didalam suatu **class (atribut & method)** menjadi **tersembunyi** dan **tidak mudah untuk diakses** oleh object-object lain **dari luar class** tersebut dengan alasan keamanan
- ❑ Penerapan konsep ini didalam pemrograman adalah kita dapat memberikan **hak akses (access modifier)** pada setiap member didalam class sehingga hanya **atribut & method tertentu** yang dapat **diakses dari luar class**



Access Modifier

- ❑ **Access modifier (hak akses)** adalah kemampuan membuat class, atribut, method, dan konstruktor **untuk dapat diakses dari bagian(scope) mana** saja didalam project.
- ❑ Apabila member Suatu Class diberikan access modifier, maka informasi yang dimilikinya **akan dibatasi** hak aksesnya sesuai dengan **level akses yang diberikan**
- ❑ Di JAVA ada 4 akses level yang dapat digunakan, yaitu **public, private, protected, default (non-modifier)**

{OOP}

Tingkatan Access Modifier

- ❑ Untuk lebih dapat memahami **tingkatan hak akses** didalam JAVA, silakan perhatikan tabel berikut

Access Levels				
Modifier	Class	Package	Subclass	World
public	Y	Y	Y	Y
protected	Y	Y	Y	T
tanpa modifier	Y	Y	T	T
private	Y	T	T	T

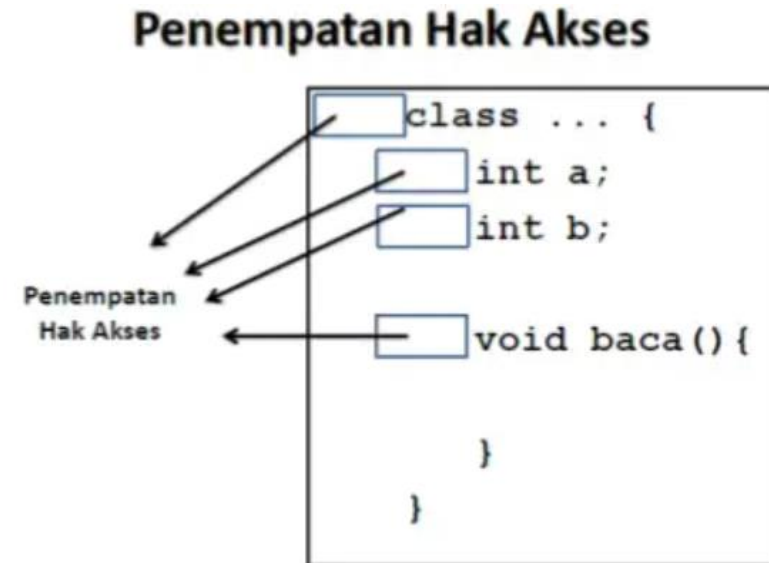
Keterangan :

Y : Bisa diakses

T : Tidak bisa diakses

Mendefinisikan Access Modifier

- ❑ Untuk dapat menambahkan access modifier pada member sebuah class (**class**, **atribut**, **method**, **constructor**), maka kita tinggal menuliskan kata kunci (**public**, **private**, **protected**) sesuai dengan level akses yang kita ingin berikan.
- ❑ Pengaturan akses terbagi menjadi 2 level,
 - Di level kelas
Pengaturan akses di level kelas terdiri dari 2 cara yaitu
 1. Tanpa menuliskan keyword apa pun (default/package-private)
 2. Menuliskan keyword **public**
 - Di level member
Pengaturan akses di level member terdiri dari 4 cara yaitu :
 1. Tanpa menulis keyword apa pun (default/package-private)
 2. Menuliskan keyword **public**
 3. Menuliskan keyword **private**
 4. Menuliskan keyword **protected**



Contoh Access Modifier pada JAVA

[Class Produk]

```
1  package tutorial3.apps;
2
3  public class Produk {
4      public String nama;
5      public int harga;
6      public int jumlah;
7
8      public Produk() {
9          //Empty
10     }
11
12     public Produk(String nama, int harga) {
13         this.nama = nama;
14         this.harga = harga;
15     }
16
17     public void tampilProduk() {
18         System.out.println("Nama Barang      : " +this.nama);
19         System.out.println("Harga Barang   : Rp." +this.harga);
20         System.out.println("Stok Saat ini  : " +this.jumlah + " Pcs");
21     }
22 }
23
24
25
```

[Class Main]

```
6  public static void main(String[] args) {
7      // TODO code application logic here
8
9      Produk produk1 = new Produk("Gula Pasir", 10000);
10     produk1.jumlah = 20;
11
12     Produk produk2 = new Produk("Minyak Goreng", 48000);
13     produk2.jumlah = 10;
14
15     System.out.println("Rekap Laporan Data Barang UNIKOM MART");
16     System.out.println("=====");
17     System.out.println("Barang ke-1");
18     produk1.tampilProduk();
19     System.out.println("\nBarang ke-2");
20     produk2.tampilProduk();
21
22 }
23
24
```

*Jika tingkat aksesnya adalah public, maka member suatu class **dapat dipanggil** oleh **objectnya dari class manapun**

BAGAIMANA DENGAN PRIVATE ??

*Dapat Anda lihat pada kelas di contoh ini , pada **pendefinisian Class, Atribut, Method, dan constructor** menggunakan tingkat akses **public**

Contoh Access Modifier pada JAVA(2)

[Class Produk]

```
1 package tutorial3.apps;
2
3 public class Produk {
4     private String nama;
5     private int harga;
6     private int jumlah;
7
8     public Produk() {
9         //Empty
10    }
11
12    public Produk(String nama, int harga){
13        this.nama = nama;
14        this.harga = harga;
15    }
16
17    public void tampilProduk() {
18        System.out.println("Nama Barang      : " +this.nama);
19        System.out.println("Harga Barang      : Rp." +this.nama);
20        System.out.println("Stok Saat ini     : " +this.jumlah + " Pcs");
21    }
22 }
23
24 }
```

[Class Main]

```
6 public static void main(String[] args) {
7     jumlah has private access in Produk
8     ----
9     (Alt-Enter shows hints)
10    new Produk("Gula Pasir", 10000);
11    produk1.jumlah = 20;
12
13    Produk produk2 = new Produk("Minyak Goreng", 48000);
14    produk2.jumlah = 10;
15
16    System.out.println("Rekap Laporan Data Barang UNIKOM MART");
17    System.out.println("=====");
18    System.out.println("Barang ke-1");
19    produk1.tampilProduk();
20    System.out.println("\nBarang ke-2");
21    produk2.tampilProduk();
22 }
```

Terjadi **ERROR** pada pemanggilan atribut oleh object

*Coba kita ganti hak akses untuk atribut pada class Produk menjadi **private**

Didalam pembuatan program dengan paradigma Object disarankan untuk selalu memberikan hak akses **PRIVATE** pada **ATRIBUT** didalam Class – (Menerapkan Enkapsulasi)

SOLUSINYA ??

Buatlah Method **Getter & Setter**

Getter dan Setter

- ❑ Pemberian tingkat akses **private** pada atribut menyebabkan nilai didalam atribut **tidak dapat diakses atau diubah** dari **luar classya**
- ❑ Agar bisa diubah, Maka kita perlu menyediakan method untuk mengubah dan mendapatkan nilai atribut tersebut yaitu **Getter** dan **Setter** [Standarisasi di JAVA]
- ❑ **Getter** adalah method untuk **mengambil data/nilai** dari suatu atribut didalam class
- ❑ **Setter** adalah method untuk **mengubah data/nilai** dari suatu atribut didalam class

{OOP}

Membuat Getter dan Setter

- ❑ **Standar** Pembuatan method getter dan setter di JAVA umumnya menggunakan **kaidah/aturan** berikut :

Tipe Data	Getter Method	Setter Method
Boolean	isNamaAtribut()	setNamaAtribut(Boolean value)
Primitif	getNamaAtribut()	setNamaAtribut(Primitif value)
Object	getNamaAtribut()	setNamaAtribut(Object value)

Sintaks Umum Getter :

```
public <tipeDataAtribut> getNamaAtribut() {  
    return <namaAtribut>;  
}
```

Sintaks Umum Setter :

```
public void setNamaAtribut(<parameterAtribut>) {  
    this.<namaAtribut> = <parameterAtribut>;  
}
```

Contoh Getter & Setter Pada JAVA

```
19 //Deklarasi Getter dan Setter Class Produk
20 public String getNama() {
21     return nama;
22 }
23
24 public void setNama(String nama) {
25     this.nama = nama;
26 }
27
28 public int getHarga() {
29     return harga;
30 }
31
32 public void setHarga(int harga) {
33     this.harga = harga;
34 }
35
36 public int getJumlah() {
37     return jumlah;
38 }
39
40 public void setJumlah(int jumlah) {
41     this.jumlah = jumlah;
42 }
```

Beberapa hal yang harus **Kita perhatikan** dalam membuat getter & setter di JAVA :

- ❑ Setiap atribut dengan **access modifier private** wajib dibuatkan getter & setternya (sepasang)
- ❑ getter & setter wajib diberikan **access modifier public**
- ❑ **getter** berupa method yang dapat membalikan nilai (**return value**)
- ❑ **setter** berupa method yang tidak membalikan nilai (**void**)

Contoh Getter & Setter Pada JAVA

Ketika kita menerapkan **prinsip enkapsulasi** didalam class (private atribut, getter & setter) , maka cara kita untuk **pengkasesan & pengisian nilai atribut dari object akan berubah**

```
5 public static void main(String[] args) {  
6     // TODO code application logic here  
7  
8     Produk produk1 = new Produk("Gula Pasir", 10000);  
9     produk1.setJumlah(20); -> Mengisi nilai jumlah menggunakan method setter  
10  
11     Produk produk2 = new Produk("Minyak Goreng", 48000);  
12     produk2.setJumlah(10); -> Mengisi nilai jumlah menggunakan method setter  
13  
14     System.out.println("Rekap Laporan Data Barang UNIKOM MART");  
15     System.out.println("=====");  
16     System.out.println("Barang ke-1");  
17     produk1.tampilProduk();  
18     System.out.println("\nBarang ke-2");  
19     produk2.tampilProduk();  
20  
21 }
```

Seperti contoh ini , nantinya kita wajib menggunakan **method getter & setter** untuk mengakses dan mengisi nilai dari atribut yg private

Terakhir mari kita lihat bagaimana memasukkan nilai pada console melalui keyboard (inputan user) di **JAVA**

Menerima Input Dari Keyboard

- ❑ Pada aplikasi berbasis console di JAVA, ada 2 mekanisme umum yang dapat digunakan untuk membaca suatu masukan dari keyboard, yaitu :

- 1. Menggunakan Class Scanner**
- 2. Menggunakan Class BufferedReader**



- ❑ Di pertemuan ini kita akan mencoba praktekkan hanya untuk pemanfaatan dari class scanner saja. (Langkah implementasi didalam program lebih mudah dari class BufferedReader)

Menerima Input Dari Keyboard

- ❑ Langkah pertama yang harus kita lakukan ada **mengimport class scanner** dari package **java.util**
- ❑ Langkah kedua **buat object** dari Class Scanner

```
1 package tutorial3.apps;
2 import java.util.Scanner; -> Langkah 1, Import class Scanner
3
4
5 public class Main {
6
7     public static void main(String[] args) {
8         // TODO code application logic here
9         Scanner inputText = new Scanner(System.in); -> Langkah 2, buat object. Isi argument
10                                                    constuktornya dengan System.in
11
12         Produk produk1 = new Produk("Gula Pasir", 10000);
13         produk1.setJumlah(20);
14
15         Produk produk2 = new Produk("Minyak Goreng", 48000);
16         produk2.setJumlah(10);
```


Menerima Input Dari Keyboard

- ❑ Langkah ketiga, **Panggil method next** dari object yg sebelumnya kita buat dari Class Scanner sesuai dengan tipe data yang akan diterima (contoh : jika menerima int gunakan method nextInt(), jika menerima String gunakan nextLine(), dst

```
public static void main(String[] args) {  
    // TODO code application logic here  
    Scanner inputText = new Scanner(System.in);
```

```
    Produk produk1 = new Produk("Gula Pasir", 10000);  
    System.out.print("Masukan Jumlah stok produk "+produk1.getNama()+" :" );  
    produk1.setJumlah(inputText.nextInt());
```

-> Langkah 3, panggil method nextInt dari object Scanner. Masukan nilainya kedalam atribut jumlah

```
    Produk produk2 = new Produk("Minyak Goreng", 48000);  
    produk2.setJumlah(10);  
  
    System.out.println("Rekap Laporan Data Barang UNIKOM MART");  
    System.out.println("=====");  
    System.out.println("Barang ke-1");  
    produk1.tampilProduk();  
    System.out.println("\nBarang ke-2");  
    produk2.tampilProduk();  
}
```

*Jika nilai yang dibaca hanya 1 tipe data saja. Cukup buat 1 object. Namun jika lebih sebaiknya buat object per tipe data

nextByte()	byte
nextByte(int i)	byte
nextInt()	int
nextInt(int i)	int
nextShort()	short
nextShort(int i)	short
next()	String
next(Pattern ptrn)	String
next(String string)	String
nextBigDecimal()	BigDecimal
nextBigInteger()	BigInteger
nextBigInteger(int i)	BigInteger
nextBoolean()	boolean
nextDouble()	double
nextFloat()	float
nextLine()	String
nextLong()	long

Contoh Kode (Program Lengkap)

```
1 package tutorial3.apps;
2
3 public class Produk {
4     //Deklarasi Atribut Class Produk
5     private String nama;
6     private int harga;
7     private int jumlah;
8
9     //Deklarasi Konstruktor
10    public Produk() {
11        //Empty
12    }
13
14    public Produk(String nama, int harga){
15        this.nama = nama;
16        this.harga = harga;
17    }
18
19    //Deklarasi Getter dan Setter Class Produk
20    public String getNama() {
21        return nama;
22    }
```

```
23
24    public void setNama(String nama) {
25        this.nama = nama;
26    }
27
28    public int getHarga() {
29        return harga;
30    }
31
32    public void setHarga(int harga) {
33        this.harga = harga;
34    }
35
36    public int getJumlah() {
37        return jumlah;
38    }
39
40    public void setJumlah(int jumlah) {
41        if(jumlah >= 0){
42            this.jumlah = jumlah;
43        }
44    }
45
46    //Deklarasi Method Class Produk
47    public void tampilProduk(){
48        System.out.println("Nama Barang : " +this.nama);
49        System.out.println("Harga Barang : Rp." +this.harga);
50        System.out.println("Stok Saat ini : " +this.jumlah);
51    }
52
53
54
55 }
```

Contoh Kode (Program Lengkap)

```
1  package tutorial3.apps;
2  import java.util.Scanner;
3
4
5  public class Main {
6
7      public static void main(String[] args) {
8          // TODO code application logic here
9          Scanner inputText = new Scanner(System.in);
10
11          Produk produk1 = new Produk("Gula Pasir", 10000);
12          System.out.print("Masukan Jumlah stok produk "+produk1.getNama()+" :" );
13          produk1.setJumlah(inputText.nextInt());
14
15          Produk produk2 = new Produk("Minyak Goreng", 48000);
16          System.out.print("Masukan Jumlah stok produk "+produk2.getNama()+" :" );
17          produk2.setJumlah(inputText.nextInt());
18
19          System.out.println("Rekap Laporan Data Barang UNIKOM MART");
20          System.out.println("=====");
21          System.out.println("Barang ke-1");
22          produk1.tampilProduk();
23          System.out.println("\nBarang ke-2");
24          produk2.tampilProduk();
25
26      }
27 }
```

Contoh Kode (Program Lengkap)

```
run:
Masukan Jumlah stok produk Gula Pasir :27
Masukan Jumlah stok produk Minyak Goreng :8
Rekap Laporan Data Barang UNIKOM MART
=====

Barang ke-1
Nama Barang      : Gula Pasir
Harga Barang     : Rp.10000
Stok Saat ini    : 27

Barang ke-2
Nama Barang      : Minyak Goreng
Harga Barang     : Rp.48000
Stok Saat ini    : 8
BUILD SUCCESSFUL (total time: 42 seconds)
```

Setelah memahami **Materi Hari ini** pada JAVA,
mari kita berlatih

***Silakan buka modul praktikum pertemuan 4**

Terima Kasih