PERTEMUAN 15
ENCAPSULATION, PACKAGE

A. TUJUAN PEMBELAJARAN

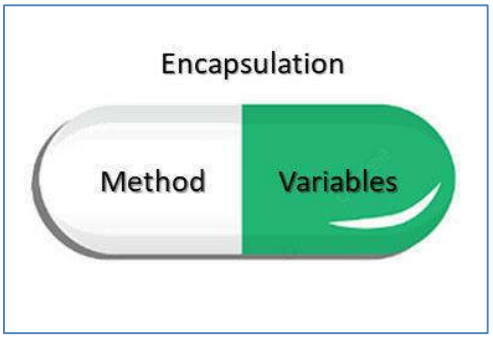
1. Mahasiswa memahami apa itu *encapsulation, package*
2. Mahasiswa memahami kelebihan dari *encapsulation*
3. Mahasiswa dapat mengetahui cara membuat package
4. Mahasiswa dapat mempraktekan *encapsulation* dan *package*

B. URAIAN MATERI

Encapsulation atau pengkapsulan adalah bagian dari konsep OOP selain dari *Inheritance*, polymorphism dan abstract, konsep ini mengikat atau membungkus sebuah objek state (field) dan *behaviour* (*method*) bersama-sama. Semua state (field) kita set atau dibungkus dengan enkapsulasi dan dibuat private agar tidak bisa di akses oleh *class* lain.

Pada Bahasa Java, *encapsulation* di implementasi melalui *access modifiers* - public, private, and protected.

Enkapsulasi pada contoh dunia nyata seperti Kapsul, yang dibungkus dengan obat-obatan yang berbeda. semua obat dikemas di dalam sebuah kapsul.



Gambar 15. 1 Ilustrasi Encapsulation

1. Cara mengimplementasikan enkapsulasi pada java :

a. Jadikan *instantance* variabel *private* sehingga tidak dapat diakses langsung dari luar *class*. Kita hanya dapat mengatur dan mendapatkan nilai dari variabel-variabel ini melalui method class.

b. Memiliki method *getter* dan *setter* di class untuk mengatur dan mendapatkan nilai *fields*.

Contoh program encapsulation pada java

```
public class enkapsulasiDemo {

    private int IDPegawai;

    private String NamaPegawai;

    private int Usia;


    //Getter and Setter methods

    public int getIdPegawai(){

        return IDPegawai;

    }

    public String getNamaPegawai(){

        return NamaPegawai;

    }


    public int  GetUsia(){

        return Usia;

    }


    public void setIdPegawai(int newValue){

    IDPegawai=newValue;

    }


    public void setNamaPegawai(String newValue){

    NamaPegawai=newValue;

    }
```

```
    public void setUsia(int newValue){
    Usia=newValue;
    }
   }
```

```
public class EncapsulasiTesting {
```

```
     public static void main(String[] args) {
        // TODO code application logic here
        enkapsulasiDemo objek=new enkapsulasiDemo();
        objek.setIdPegawai(11111);
        objek.setNamaPegawai("Agustav");
        objek.setUsia(30);

        System.out.println("Id   Pegawai   :   "   +
objek.getIdPegawai());
        System.out.println("Nama   Pegawai   :   "   +
objek.getNamaPegawai());
        System.out.println("Usia      :      "      +
objek.GetUsia());

    }
  }
```

Output :

```
Id Pegawai : 11111
Nama Pegawai : Agustav
Usia : 30
```

Pada contoh di atas, ketiga anggota data (atau field) bersifat private (lihat: *Access modifier* di java) yang tidak dapat diakses secara langsung. Field ini hanya dapat diakses melalui metode public. Fields NamaPegawai, Idpegawai dan Usia dibuat menjadi hidden data field menggunakan teknik enkapsulasi OOP.

2. Keuntungan menggunakan enkapsulasi:

Encapsulation meningkatkan pemeliharaan, fleksibilitas dan re-usability (penggunaan kembali) contoh implementasi dalam kode di atas dari void setNamaPegawai(String NamaPegawai) dan String getNamaPegawai() dapat diubah kapan saja. Karena implementasinya murni tersembunyi untuk diluar *class*, mereka masih akan mengakses field private NamaPegawai menggunakan metode yang sama setNamaPegawai(String NamaPegawai) dan getNamaPegawai()). Oleh karena itu kode dapat dipertahankan kapan saja tanpa merusak class yang menggunakan kode tersebut. Ini meningkatkan kegunaan kembali class yang mendasarinya.

Field dibuat hanya dapat dibaca (Jika kita tidak mendefinisikan metode setter di class) atau hanya tulisi (Jika kita tidak mendefinisikan metode getter di class). Untuk itu Jika kami memiliki field (atau variabel) yang tidak ingin di ubah, jadi cukup mendefinisikan variabel sebagai private dan alih-alih men setter dan getter keduanya, kita hanya perlu mendefinisikan metode get untuk variabel itu. Karena metode set tidak ada, tidak mungkin class luar dapat mengubah nilai field tersebut.

Enkapsulasi juga dikenal sebagai "Menyembunyikan data"

Dengan hanya menyediakan method setter atau getter, kita dapat membuat class hanya read(baca) atau write(tulis) saja. Dengan kata lain, kita dapat melewati method Get atau Set.

Misalkan kita ingin menetapkan nilai idPegawai 11111 saja, kita dapat menulis logika di dalam method setter.

Ini adalah cara untuk menyembunyikan data di Java karena class lain tidak akan dapat mengakses data melalui anggota private.

Contoh 2 program encapsulation pada java

Mari kita lihat contoh enkapsulasi lain yang hanya memiliki empat field dengan method setter dan getter.

```
public class account {
private long no_acc;
private String nama,email;
private float jumlah;
//public getter and setter methods
public long getAcc_no() {
    return no_acc;
}
public void setAcc_no(long no_acc) {
    this.no_acc = no_acc;
}
public String getName() {
    return nama;
}
public void setName(String nama) {
    this.nama = nama;
}
public String getEmail() {
    return email;
}
public void setEmail(String email) {
    this.email = email;
}
```

```
public float getAmount() {
    return jumlah;
}
public void setJumlah(float jumlah) {
    this.jumlah = jumlah;
}

}
```

```
public class EnkapsulasiTest2 {

    /**
     * @param args the command line arguments
     */
    public static void main(String[] args) {
        // TODO code application logic here
        account objek=new account();
        objek.setAcc_no(070123);
        objek.setName("Agustav");
        objek.setEmail("agustav@gmail.com");
        objek.setJumlah(50000);

        System.out.print(objek.getAcc_no()+"
"+objek.getName()+" "+objek.getEmail()+"
"+objek.getAmount());

    }
    }
```

Output :

```
    1070123    Agustav    agustav@gmail.com    50000.0
```

3. Package pada Java

Package seperti namanya adalah paket (grup) dari sebuah *class, interface*, dan paket lainnya. Pada java kita menggunakan *Package* untuk mengatur class dan interface. *Package* memiliki dua jenis yaitu *Built-int Package* atau bawaan dan *Package* dapat kita buat (juga dikenal sebagai package *user define*). Dalam pertemuan ini kita akan mempelajari apa itu package, apa itu package yang ditentukan pengguna dan bagaimana menggunakannya.

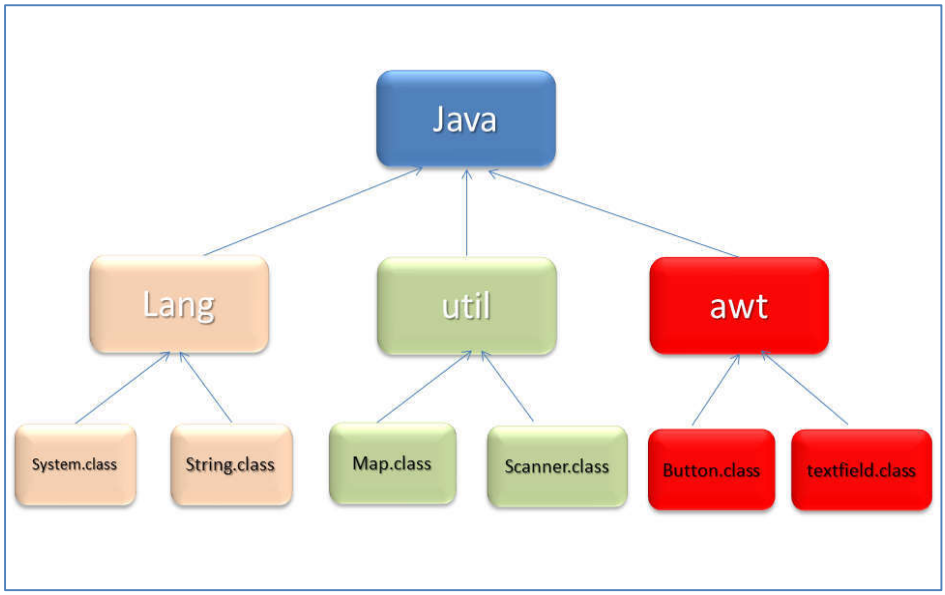
Pada java kita memiliki beberapa *package* bawaan, misalnya ketika kita membutuhkan input pengguna, kita mengimpor package seperti ini:

```
import java.util.Scanner
```

→ **java** adalah top *level package*

→ **util** adalah *sub package*

→ dan ***Scanner*** adalah *class* yang ada pada *sub package* **util**.



Gambar 15. 2 ilustrasi *package java*

Sebelum kita mempelajari cara membuat package yang ditentukan pengguna /*user define*, mari kita lihat keuntungan menggunakan package.

4. Keuntungan menggunakan Package

Inilah alasan mengapa kita harus menggunakan paket di Java:

a. Re-usability (Dapat digunakan kembali): Saat mengembangkan proyek di java, kita sering merasa bahwa ada kode program yang kita tulis berulang kali. dengan menggunakan package, kita dapat membuat hal-hal seperti itu dalam bentuk class di dalam pacakage dan kapan pun kita perlu melakukan tugas yang sama, cukup mengimpor package dan menggunakan class tersebut.
b. Organization (Peng organisasian yang lebih baik): Dalam project java yang besar di mana kita memiliki beberapa ratus class, selalu diperlukan untuk mengelompokkan jenis kelas yang serupa dalam nama package yang bermakna sehingga kita dapat mengatur project dengan lebih baik dan ketika kita membutuhkannya salah satu class, kita dapat dengan cepat menemukan dan menggunakannya, sehingga meningkatkan efisiensi.
c. Menghindari Konflik Nama: Kita dapat mendefinisikan dua class dengan nama yang sama dalam paket yang berbeda sehingga untuk menghindari tabrakan nama, kita dapat menggunakan package.

5. Jenis jenis Package

Seperti yang disebutkan di awal pembahasan yaitu pacakge memiliki dua jenis pada java.

a. Package yang ditentukan pengguna: Package yang kita buat disebut package yang ditentukan pengguna (User Defined).
b. Package bawaan: Package yang sudah ditentukan seperti java.io.*, java.lang.* dll dikenal sebagai Package bawaan.

Contoh 1. program menggunakan package:

```
. package kalkulatorpack;


/**
 *
 * @author feiruz
 */
public class kalkulator {
     public int pertambahan(int a, int b){
      return a+b;
}
       public static void main(String args[]){
      kalkulator obj = new kalkulator();
      System.out.println(obj.tambah(10, 20));
   }
}
```

Kita telah membuat *class* Kalkulator di dalam nama package **kalkulatorpack**. Untuk membuat kelas di dalam sebuah package, nyatakan nama paket dalam pernyataan pertama dalam program Anda. Sebuah kelas hanya dapat memiliki satu deklarasi package.

File kalkulator.java dibuat di dalam package **kalkulatorpack**

Sekarang mari kita lihat bagaimana menggunakan package ini di program lain.

```
import kalkulatorpack.kalkulator;
public class kalkulatorTambah {



    /**
     * @param args the command line arguments
```

```
     */
    public static void main(String[] args) {
        // TODO code application logic here
    kalkulator objek=new kalkulator();
    System.out.println(objek.pertambahan(10,30));

   }

}
```

Output :

```
      40
```

Dalam contoh program di atas terlihat Untuk menggunakan *class* Kalkulator, kita impor package **kalkulatorpack.** import kalkulatorpack.kalkulator, ini hanya mengimpor kelas kalkulator. Namun jika kita memiliki beberapa class di dalam package **kalkulatopack** maka kita dapat mengimpor package menggunakan semua class dari package dengan menulisi seperti dibawah ini.

```
import kalkulatorpack.*
```

C. LATIHAN / TUGAS

1. Modifikasi contoh 1 *package* **kalkulatorpack** dengan menambahkan pada class kalkulator method methodnya yaitu :
   a. Method pengurangan (int a,int b)
   b. Method pembagian (float a,float b)
   c. Method perkalian (float a,float b)
   d. Berikan masing masing parameternya
2. Buat class untuk masing masing nya :dan import package ***kalkulatorpack*** yang telah dibuat

a. KalkulatorKurang.java
b. KalkulatorBagi.java
c. KalkulatorKali.java

3. Tampilkan hasil output nya

D. REFERENSI

Horstmann Cay S., (2011). *Big Java 4t^{h} Edition ,san jose university , united state Of America*. *RRD jefferson city publishing.*

Deitel Paul , Deitel Harvey, (2012) Java how to program eighth edition, pearson education, Boston Massachusetts , *USA, publishing as prentice hall.*

Rose Cristhoper, (2017), Java Succinctly Part 2, Morrisville, NC 27560, USA, Syncfusion, Inc.

Downey Allen B. , Mayfield Chris, (2017), Think Java, Needham, Massachusetts, USA, Green Tea Press

Hayes Helen, (2021), BeginnersBook.com, https://beginnersbook.com/java-tutorial-for-beginners-with-examples/, di akses pada tanggal 21 November 2021.

Sonoo Jaiswal , (2021) JavaTpoint offers college campus training , https://www.javatpoint.com/java-tutorial, diakses pada tanggal 1 Desember 2021