Pemrograman Berorientasi Obyek

# Encapsulation & Inheritance

Ramos Somya

# Encapsulation

- Adalah pembungkusan attribute atau behaviour sehingga tidak dapat diganti secara sembarangan dengan cara yang tidak seharusnya
- Adalah sebuah konsep di mana data dan method / prosedur / function dibungkus ke dalam sebuah wadah yang disebut dengan objek.
- Adalah sebuah mekanisme untuk me-restrict pengaksesan terhadap sebuah komponen dari objek.

# Manfaat Encapsulation

- Information Hiding: Karena kita dapat menentukan hak akses (public, private, protected, default) sebuah variabel/method dari objek, dengan demikian kita bisa menyembunyikan informasi yang tidak perlu diketahui objek lain.
- Bisa juga digunakan untuk validasi.

# Contoh

```
public class Kucing {

    String nama;
    String warna;
    int umur;
    int berat;
```

**Dapat diakses langsung tanpa adanya kontrol**

- Untuk membatasi akses (Visibility) → **Access Modifier.**

# Access Modifier

- Adalah sebuah kata kunci / keyword yang digunakan untuk menentukan hak akses kelas lain terhadap sebuah kelas beserta attribute dan behaviour dari kelas tersebut.
- Dituliskan di depan kelas / field / method.
- Ada 3 yang umum:
  - protected
  - private
  - public

# public

- Dapat diakses dari dalam dan luar kelas.
- Menggunakan kata kunci “public”.
- Dapat di-apply pada kelas, field dan method.

# private

- Hanya dapat diakses dari dalam kelas itu sendiri.
- Menggunakan kata kunci “private”.

# protected

- Hanya dapat diakses dari kelas itu dan kelas – kelas yang merupakan turunan dari kelas tersebut.
- Menggunakan kata kunci “protected”.

# default

- Hanya kelas – kelas dalam package yang sama yang memiliki hak akses terhadap attribut dan method dalam class.
- Jenis ini tidak memiliki kata kunci.
- Semua atribut dan method yang tidak secara eksplisit dituliskan access modifier nya dianggap default.
- Dapat di-apply pada kelas, field dan method.

# Field Encapsulation

- Field encapsulation adalah salah satu teknik OOP untuk menghindarkan pengaksesan secara langsung terhadap isi dari attribute / field sebuah kelas tertentu dari kelas lainnya.
- Setiap field yang akan dienkapsulasi diberi access modifier private sehingga tidak dapat diakses dari luar kelas tersebut.
- Bagaimana mengaksesnya??? Menggunakan getter setter.
- Dalam OOP, getter dan setter merupakan implementasi dari enkapsulasi.

# Getter dan Setter

- Getter (accessor) adalah sebuah method yang digunakan untuk mengambil nilai / isi sebuah attribute kelas yang bersifat private.

- Ciri getter:
  - Memiliki access modifier yang dapat diakses dari luar kelas tersebut
  - Memiliki return value
  - Nama method diawali dengan "get" diikuti nama attribute yang diawali huruf besar (umumnya)

- Setter (mutator) adalah sebuah method yang digunakan untuk mengisi nilai / isi sebuah attribute kelas yang bersifat private.

- Ciri setter:
  - Memiliki access modifier yang dapat diakses dari luar kelas tersebut
  - Memiliki sebuah parameter sebagai value untuk attribute yang diset
  - Nama method diawali dengan "set" diikuti nama attribute yang diawali huruf besar (umumnya)

# Contoh

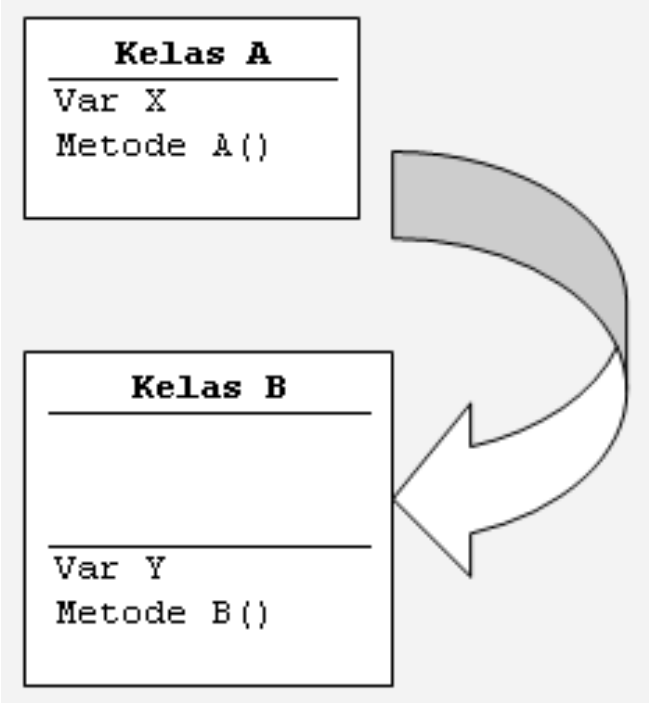
```
public class Kucing {

    private String nama;
    private String warna;
    private int umur;
    private int berat;

    public int getBerat() {
        return berat;
    }

    public void setBerat(int berat) {
        this.berat = berat;
    }

    public String getNama() {
        return nama;
    }

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

```
    public int getUmur() {
        return umur;
    }

    public void setUmur(int umur) {
        if (umur > 10) {
            System.out.println("Umur tidak boleh dari 10");
        } else {
            this.umur = umur;
        }
    }

    public String getWarna() {
        return warna;
    }

    public void setWarna(String warna) {
        this.warna = warna;
    }

    void Meong() {
        System.out.println("Metoda Meong");
    }

    void Gigit() {
        System.out.println("Metoda Gigit");
    }
}
```

```
public class Tester {

    public static void main(String[] args) {
        Kucing garong = new Kucing();

        garong.setNama("Si Garong");
        garong.setWarna("Hitam");
        garong.setUmur(14);
        garong.setBerat(3);

        System.out.println("Nama : " + garong.getNama());
        if (garong.getUmur() > 0 && garong.getUmur() < 10) {
            System.out.println("Umur : " + garong.getUmur());
        }
        garong.Meong();
        garong.Gigit();

    }
}
```

# Inheritance / Pewarisan

- Pewarisan attribute dan behaviour sebuah kelas kepada kelas yang lain.
- Kelas turunan biasa disebut dengan *child class / subclass* sedangkan kelas yang mewarisi biasa disebut dengan *parent class / superclass*.
- Selain mewarisi *state* dan *behaviour* dari *superclass*-nya, *subclass* kemudian dapat menambahkan *state* dan *behaviour* baru yang spesifik.




- Kelas A disebut Super Class(Parent Class).
- Kelas B disebut Sub Class(Child Class).

# Cara pewarisan

```
class KelasTurunan extends KelasDasar{
    tubuh kelas
}
```

# contoh

```
public class Ayah {
    String marga = "Rajaguguk";
    public void infoMarga(){
        System.out.println("Marga = "+this.marga);
    }

    public void hidungMancung(){
        System.out.println("Bentuk hidung mancung");
    }
}
```

...

```
public class Anak extends Ayah {
    String hobi;
}
```

...

```
public class TestPewarisan {

    public static void main(String[] args) {
        Anak poltak = new Anak();
        poltak.infoMarga();
        poltak.hidungMancung();

    }
}
```

# contoh

```
public class Ayah {

    private String nama;
    protected String marga = "Rajaguguk";
    public void infoMarga(){
        System.out.println("Marga = "+this.marga);
    }

    public void hidungMancung(){
        System.out.println("Bentuk hidung mancung");
    }
}
```

```
public class Anak extends Ayah {
    String hobi;
}
```

- Pada kelas turunan (Anak) kita tidak dapat mengakses variabel yang memiliki akses modifier berjenis private yaitu nama yang terdapat pada kelas dasar (Ayah).

- Agar dapat diakses hanya oleh kelas turunan saja, kita dapat mengganti akses modifier variabel nama yang semula private menjadi protected.

# Latihan 1

- Buat KELAS Mahasiswa dengan atribut private NIM, Nama, Nilai, serta fungsi Indeks yang mengimplementasikan aturan pencetakan : Jika Nilai > 60 → Lulus dan jika Nilai <= 60 → Tidak Lulus.
- Buat setter dan getter untuk NIM, Nama, serta Nilai.
- Buat KELAS Dosen sedemikian rupa sehingga bisa memanfaatkan KELAS Mahasiswa dan mengisinya dengan nilai-nilai atribut sesuka Anda, serta mencetak keterangan Indeks berdasarkan nilai yang dimasukkan tadi.

# Latihan 2

- Buat SUPERKELAS Mahasiswa dengan data/atribut NIM, Nama, IPK.
- Buat SUBKELAS Mhs_S1, Mhs_S2, Mhs_S3, dengan metoda/fungsi apaCumlaude() yang menuliskan keterangan Cumlaude jika IPK Mhs_S1 > 3,5, jika IPK Mhs_S2 > 3,75, jika IPK Mhs_S3 > 3,9, dan menuliskan keterangan Tidak Cumlaude jika IPK di bawah nilai-nilai tadi.
- Buat KELAS Uji_Mahasiswa yang membentuk objek-objek dari kelas-kelas Mhs_S1, Mhs_S2, dan Mhs_S3, dengan nilai bebas, dan kemudian menuliskan keterangannya masing-masing.

# Tugas Take Home