Enkapsulasi

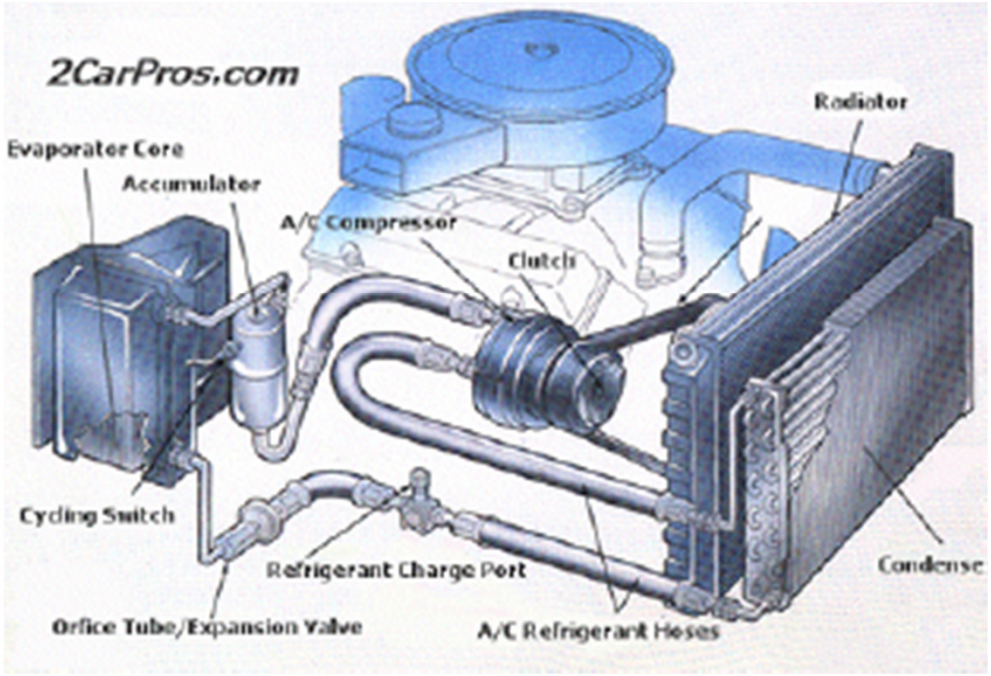
Edi Sugiarto, S.Kom, M.Kom

Pendahuluan

- Dalam pemrograman berorientasi objek istilah enkapsulasi artinya menyembunyikan data di dalam kelas.
- Data yang dimaksud adalah instance variabel yg memiliki nilai ekslusif dari kelas/objek
- Sebenarnya objek dapat dikatakan sebagai entitas yang mengikat data-data yang ekslusif.

Apa itu Enkapsulasi?

- Memisahkan aspek-aspek eksternal objek, yang dapat diakses objek-objek lain dari rincian implementasi objek yang tersembunyi dari objek-objek lain.
- Dapat diartikan sebagi bungkusan atau pelindung data yang sedang diolah, pembungkus ini mendefinisikan perilaku dan melindungi program dan data yang sedang diolah agar tidak diakses oleh objek lain



edi.sugiarto@yahoo.com

- Pada umumnya struktur dari objek adalah tersembunyi, juga implementasi dari method.
- Yang nampak hanyalah layanan-layanan yang dapat diminta dari objek atau objek lain.
- Prinsipnya enkapsulasi adalah penyembunyian informasi (information hiding).

Tujuan Enkapsulasi

- Agar program terhindar dari ketergantungan terhadap perubahan yang menyebabkan akibat berurutan/beruntun yang besar.
- Pengkapsulan meredam perubahan menjadi ke hanya objek tersebut atau sekelompok kecil objek yang memang terkait erat.

Manfaat Enkapsulasi

- **Modularitas**
  - Kode sumber sebuah objek dapat dikelola secara independen dari kode sumber objek lain
- **Information Hiding**
  - Memungkinkan objek menyembunyikan informasi yang tidak perlu diketahui objek lain

Visibility Modifier

- Merupakan modifier yang memberikan batasan kemampuan variabel atau method untuk diakses.

Beberapa visibility modifier

- Public
- Protected
- default
- Private

Class Acces Level

Specifier	Class	Package	SubClass	World
private	✓			
no specifier	✓	✓		
protected	✓	✓	✓	
public	✓	✓	✓	✓

edi.sugiarto@yahoo.com

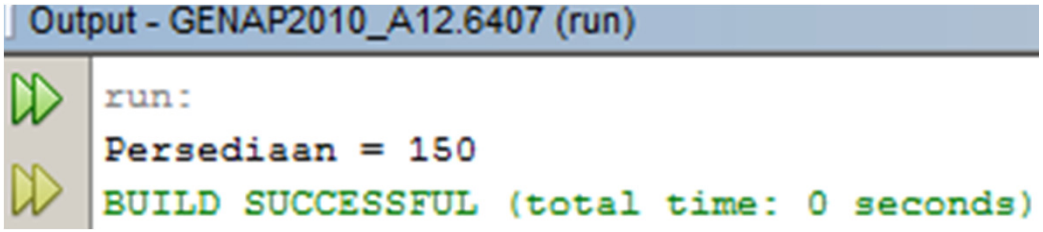
Modifier Public

- Modifier *public adalah modifier yang memberi kemampuan tak* terbatas bagi variabel atau method untuk diakses
- Artinya, variabel atau method yang menggunakan modifier *public akan dapat diakses* dari mana saja, baik dari dalam class sendiri, maupun dari class lain

Contoh

```
public class lumbungPadi {

    public int persediaan;
    public int jumlahDiambil;
    public int jumlahDimasukkan;

    public void cetakPersediaan(){
    persediaan = persediaan + jumlahDimasukkan-jumlahDiambil;
    System.out.println("Persediaan = " +persediaan);
    }

}
```

```
public class kegiatanPanen {

    public static void main(String[ ] args){
        lumbungPadi lumbungDesaSukatani;
                lumbungDesaSukatani = new lumbungPadi( );

        lumbungDesaSukatani.persediaan = 100;
        lumbungDesaSukatani.jumlahDimasukkan = 200;
        lumbungDesaSukatani.jumlahDiambil = 150;
        lumbungDesaSukatani.cetakPersediaan( );
    }

}
```



- Setiap attribut atau method dengan akses modifier public maka dapat di akses oleh objek lain dan tidak ada batasan.

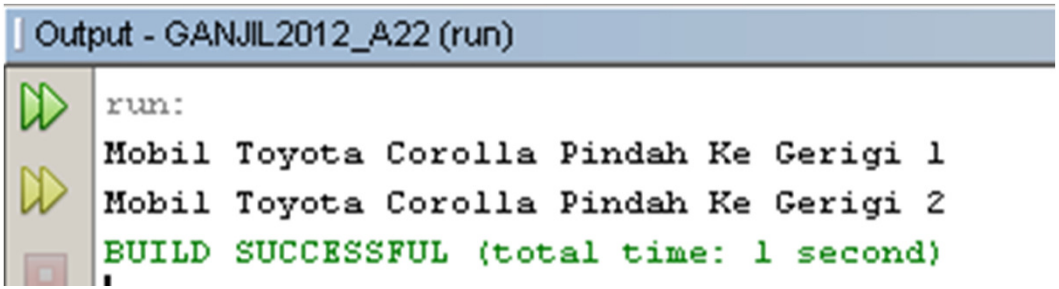
Modifier Protected

- Modifier protected memberikan kemampuan pada variabel atau method agar dapat diakses semua kelas dalam satu paket
- Attribut atau method dengan akses modifier protected tidak dapat diakses oleh kelas lain di luar paket (kecuali kelas turunan)
- Attribut atau method dengan akses modifier protected dapat diakses oleh kelas yg merupakan subclass dari kelas dimana attribut atau method tersebut ditempatkan.

Contoh

```
public class mobil {

    protected String merk="";
    protected String warna="";
    int gerigi=0;

    protected void tambahGerigi()
    {
        gerigi++;
        System.out.println("Mobil "+ this.merk +
                " Pindah Ke Gerigi "+gerigi);
    }

}
```

```
*/
public class mobil_Main {

    public static void main(String[] args)
    {
        mobil sedan = new mobil();
        sedan.merk="Toyota Corolla";
        sedan.warna="Merah";
        sedan.tambahGerigi();
        sedan.tambahGerigi();
    }
}
```



Modifier Default

- attribut atau method dengan akses modifier default maka variabel atau method tersebut dapat di akses oleh kelas lain dalam paket yang sama.
- Tidak ada keyword khusus untuk mendeklarasikan modifier default access
- Sehingga ketika attribut atau method didefinisikan tanpa akses modifier maka sebenarnya akses modifiernya adalah *default*

Contoh

```
public class mobil {

    String merk="";
    String warna="";
    int gerigi=0;

    void tambahGerigi()
    {
        gerigi++;
        System.out.println("Mobil "+ this.merk +
                " Pindah Ke Gerigi "+gerigi);
    }

}
```

```
public class mobil_Main {

    public static void main(String[] args)
    {
        mobil sedan = new mobil();
        sedan.merk="Honda Accord";
        sedan.warna="Biru";
        sedan.tambahGerigi();
        sedan.tambahGerigi();
    }
}
```

Modifier Private

- Merupakan modifier yang membatasi aksesibilitas variabel atau method, sehingga hanya dapat diakses dari kelas yang sama.
- Maka attribut atau method dengan akses modifier ini hanya bisa diakses oleh kelas itu sendiri.

Contoh

```
public class televisi {

    private String merk;
    private String warna;
    private boolean mesin=false;

    public void setMerk(String merk)
    {
        this.merk = merk;
    }
    public void setWarna(String warna)
    {
        this.warna = warna;
    }
    private void nyalakanMesin()
    {
        this.mesin=true;
        System.out.println("Mesin Televisi "+ this.merk+" Menyala");
    }
    protected void on()
    {
        nyalakanMesin();
    }

}
```

Contoh

```
public class televisi_Main {

    public static void main(String[] args)
    {
        televisi hitachi = new televisi();
        hitachi.setMerk("Hitachi");
        hitachi.setWarna("Hitam");
        hitachi.on();
    }
}
```

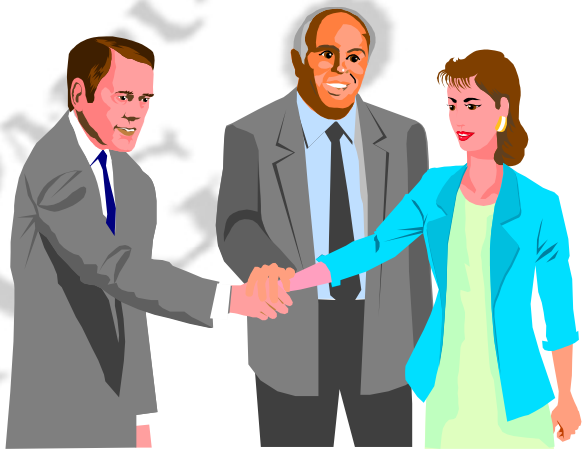
What???

Ada Pertanyaan ?

Why ???

edi.sugiarto@yahoo.com

Terima kasih



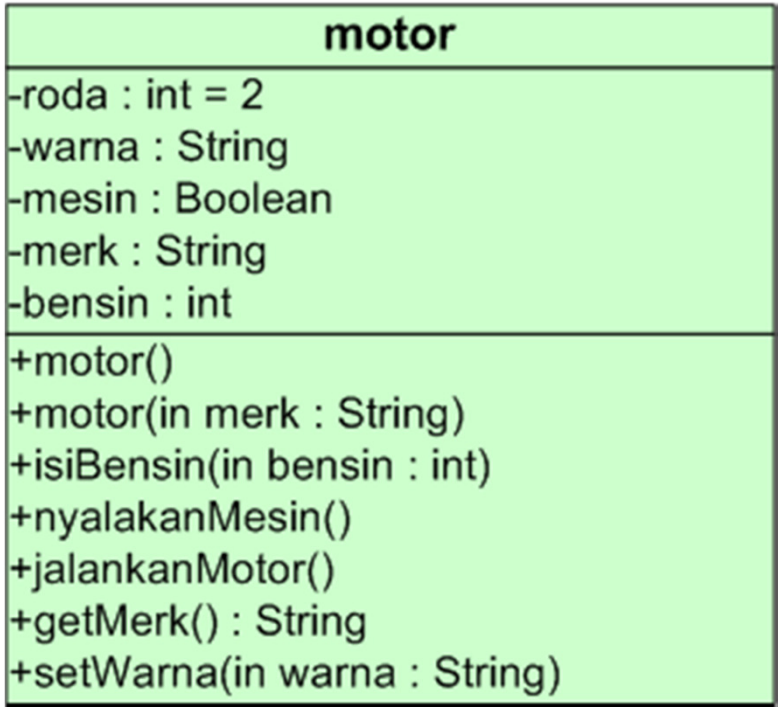
edi.sugiarto@yahoo.com

Daftar Pustaka

- Java™ Tutorial, Third Edition: A Short Course on the Basics, Addison Wesley , 2000.
- Kathy Sierra & Bert Bates, “Sun Certified Programer for Java tm 6 Study Guide”, McGraw-Hill Companies, 2008.
- Liem, I. (2003). Diktat Kuliah Pemrograman Berorientasi Objek. Departemen Teknik Informatika Institut Teknologi Bandung

Latihan 6.1

- Buatlah implementasi kelas dari kelas diagram berikut.



Petunjuk

- Motor dapat dijalankan jika mesin sudah menyala.
- Mesin dapat menyala jika terdapat bahan bakar.

Latihan 6.2

- Buat sebuah kelas yang merepresentasikan sebuah Radio, kelas ini memiliki attribut : warna, volume, baterai, mesin, dan memiliki method : isiBaterai(), nyalakan(), ubahVolume().

Petunjuk

- radio dapat diganti volume jika mesin sudah menyala,
- radio dapat menyala jika terdapat baterai,
- sehingga pengguna harus mengisikan baterai dahulu agar radio dapat digunakan