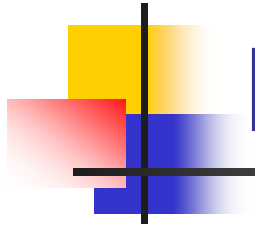




# Enkapsulasi

---

## Praktikum 7

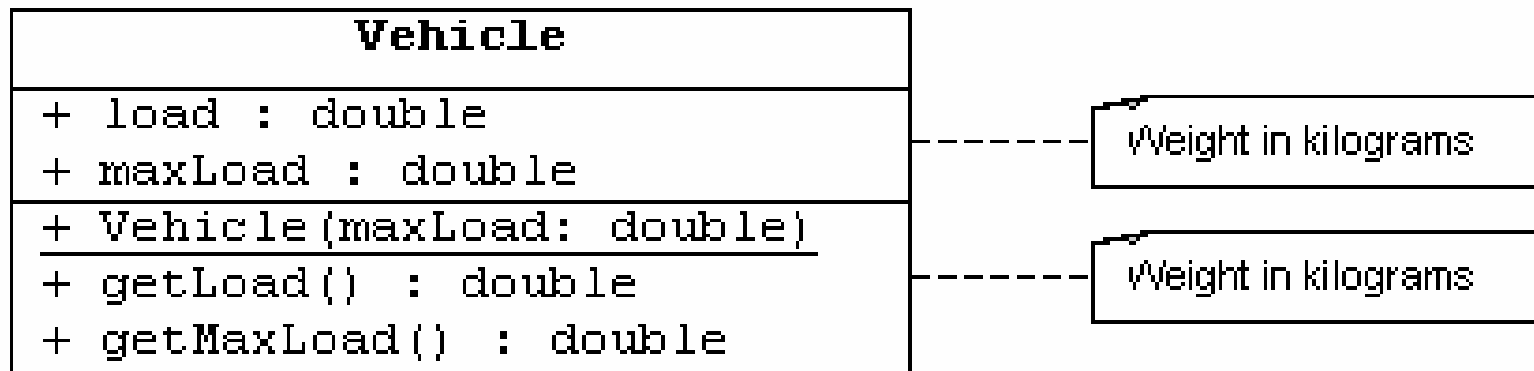


# Enkapsulasi & Abstraksi Data

---

- Tujuan :
  - mengetahui tujuan enkapsulasi dan abstraksi data.
  - Membuat kelas dalam tiga tahap yang mendemonstrasikan penggunaan information hiding.

## Versi 1: Tidak ada information hiding



Gambar 3.1 UML class diagram of Vehicle with no Hiding



## Versi 1: Tidak ada information hiding

---

Pada versi 1 ini, Vehicle class berisi atribut yang mempunyai modifier public, sehingga TestVehicle1 sebagai test program mempunyai akses langsung terhadap atribut pada Vehicle.




## Versi 1: Lakukan langkah-langkah berikut:

- Buatlah **Vehicle** class yang mengimplementasikan UML diagram yang telah diberikan pada gambar 3.1.
  - a. Tambahkan dua buah atribut yang bertipe public: **load** (the current weight of the vehicle's cargo) dan **maxLoad** (the vehicle's maximum cargo weight limit).
  - b. Tambahkan satu buah konstruktor yang bertipe public, yang digunakan untuk mengeset nilai atribut **maxLoad**.
  - c. Tambahkan dua buah methods yang bertipe public: **getLoad** (untuk mendapatkan nilai atribut **load**) dan **getMaxLoad** (untuk mendapatkan nilai atribut **maxLoad**).

Note: Semua data diasumsikan dalam satuan **kilogram**

Versi 1: Setelah selesai dengan Vehicle.java, ketik program TestVehicle.java berikut:



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

        // Create a vehicle that can handle 10,000 kilograms weight
        System.out.println("Creating a vehicle with a 10,000kg maximum load.");
        Vehicle vehicle = new Vehicle(10000.0);

        // Add a few boxes
        System.out.println("Add box #1 (500kg)");
        vehicle.load = vehicle.load + 500.0;

        System.out.println("Add box #2 (250kg)");
        vehicle.load = vehicle.load + 250.0;

        System.out.println("Add box #3 (5000kg)");
        vehicle.load = vehicle.load + 5000.0;

        System.out.println("Add box #4 (4000kg)");
        vehicle.load = vehicle.load + 4000.0;

        System.out.println("Add box #5 (300kg)");
        vehicle.load = vehicle.load + 300.0;

        // Print out the final vehicle load
        System.out.println("Vehicle load is " + vehicle.getLoad() + " kg");
    }
}
```

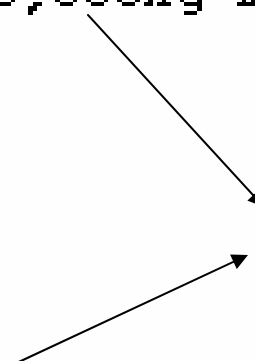


# Versi 1: tes TestVehicle.java

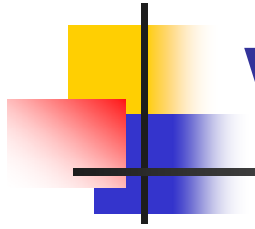
---

- Kompile Vehicle dan TestVehicle.
- Jalankan TestVehicle, seharusnya keluar tampilan sebagai berikut:

```
Creating a vehicle with a 10,000kg maximum load.  
Add box #1 (500kg)  
Add box #2 (250kg)  
Add box #3 (5000kg)  
Add box #4 (4000kg)  
Add box #5 (300kg)  
Vehicle load is 10050.0 kg  
Finished executing
```



**TROUBLE**



## Versi 1: Resume

---

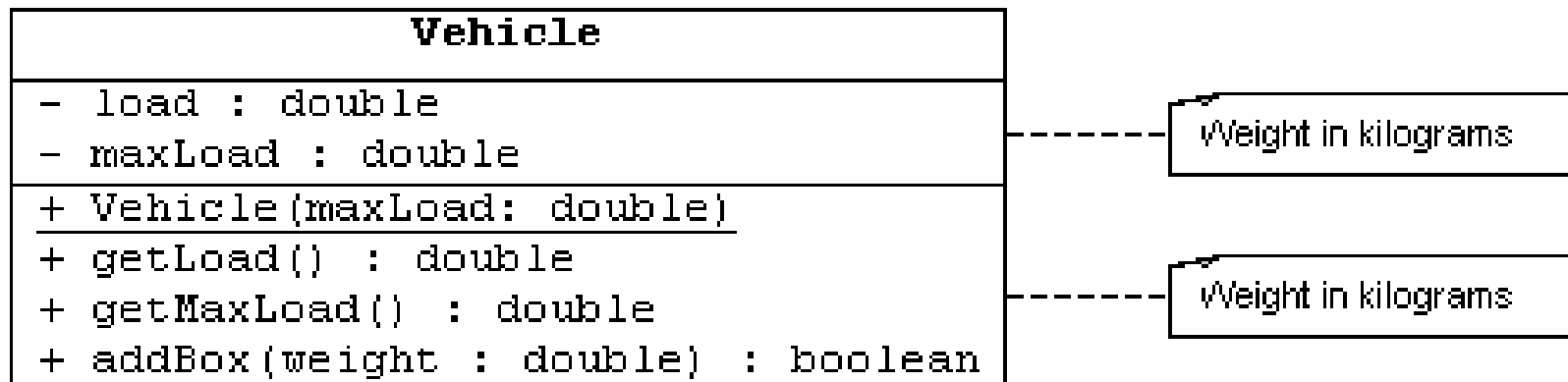
- Perhatikan bahwa pada TestVehicle, dibuat Vehicle dengan kapasitas maksimum 10.000 kg.
- Tetapi pada program selanjutnya terdapat penambahan boxes yang melebihi kapasitas (10.050 kg) → trouble.
- Kenapa trouble? Karena tidak ada pengecekan kapasitas maksimum sehingga vehicle nya kelebihan kapasitas.
- Untuk mengatasi hal ini lanjutkan ke praktikum versi 2.





## Versi 2: Dengan menggunakan information hiding

---



Gambar 3.2 UML class diagram of vehicle with information hiding



## Versi 2: Dengan menggunakan information hiding

---

- Untuk menyelesaikan masalah versi 1, sebaiknya kita menyembunyikan data internal (`load` dan `maxLoad`) dan menyediakan method, `addBox`, sebagai fasilitas pengecekan terhadap `maxLoad` supaya tidak terjadi kelebihan kapasitas.



## Versi 2: Dengan menggunakan information hiding

---

- Kopi Vehicle.java dan lakukan modifikasi untuk mengimplementasikan UML diagram pada gambar 3.2. → Vehicle1.java:
  - Lakukan modifikasi terhadap atribut `load` dan `maxLoad` → jadikan pertipe private.
  - Tambahkan method `addBox`. Method ini mempunyai satu argumen yaitu `weight` dalam satuan kilogram.  
Method `addBox` harus melakukan pengecekan terhadap penambahan box agar jangan sampai melebihi kapasitas maksimum.  
Bila terjadi pelanggaran terhadap kapasitas maksimum, maka penambahan box di tolak dan mengembalikan nilai `false`; jika tidak terjadi pelanggaran terhadap batas maksimum maka `weight` dari box diterima dan ditambahkan pada vehicle dan mengembalikan nilai `true`.

Lanjutkan →



## Versi 2: Dengan menggunakan information hiding

---

- Hint: Gunakan statement if...else untuk melakukan pengecekan terhadap kapasitas maksimum.
- Contoh:

```
if (<boolean_expression>) {  
    <statement>  
} else {  
    <statement>  
}
```



Versi 2: Setelah selesai dengan Vehicle1.java, ketik program TestVehicle1.java berikut

---

```
public class TestVehicle {  
    public static void main(String[] args) {  
  
        // Create a vehicle that can handle 10,000 kilograms weight  
        System.out.println("Creating a vehicle with a 10,000kg maximum load.");  
        Vehicle vehicle = new Vehicle(10000.0);  
  
        // Add a few boxes  
        System.out.println("Add box #1 (500kg) : " + vehicle.addBox(500.0));  
        System.out.println("Add box #2 (250kg) : " + vehicle.addBox(250.0));  
        System.out.println("Add box #3 (5000kg) : " + vehicle.addBox(5000.0));  
        System.out.println("Add box #4 (4000kg) : " + vehicle.addBox(4000.0));  
        System.out.println("Add box #5 (300kg) : " + vehicle.addBox(300.0));  
  
        // Print out the final vehicle load  
        System.out.println("Vehicle load is " + vehicle.getLoad() + " kg");  
    }  
}
```

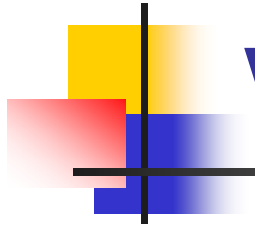


## Versi 2: tes TestVehicle1.java

---

- Kompile Vehicle1 dan TestVehicle1.
- Jalankan TestVehicle1, seharusnya keluar tampilan sebagai berikut:

```
Creating a vehicle with a 10,000kg maximum load.  
Add box #1 (500kg) : true  
Add box #2 (250kg) : true  
Add box #3 (5000kg) : true  
Add box #4 (4000kg) : true  
Add box #5 (300kg) : false  
Vehicle load is 9750.0 kg  
Finished executing  
,
```

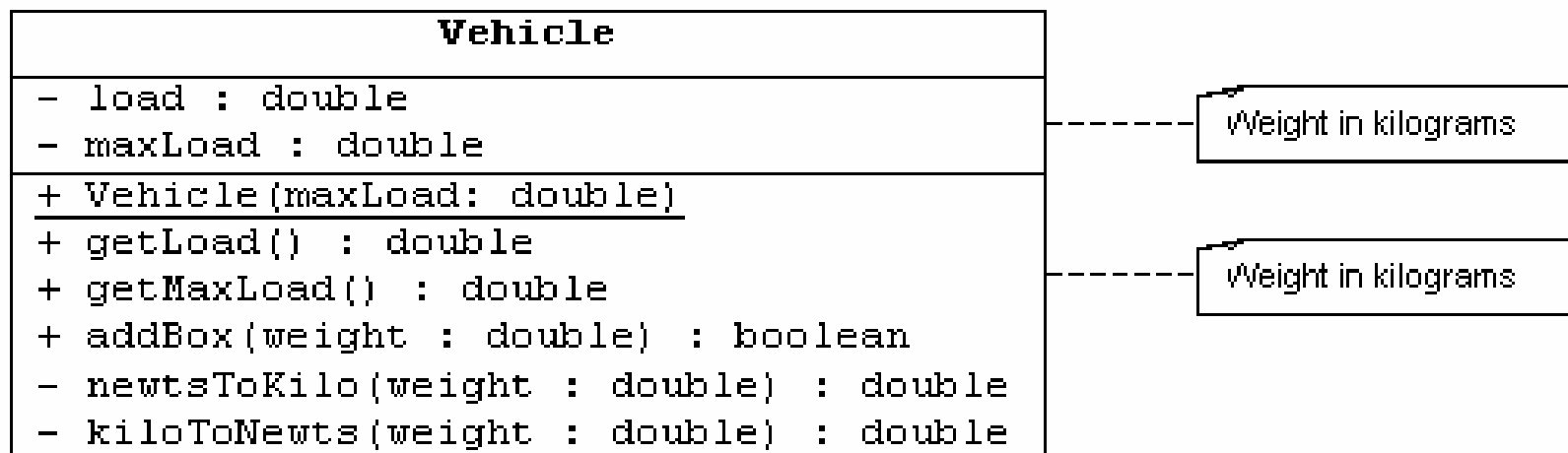


## Versi 2: Resume

---

- Pada versi 2 , pada penambahan box ke 5 terjadi kelebihan kapasitas maksimal sehingga method `addBox` mengembalikan nilai `false`, dalam arti bahwa terjadi penolakan terhadap penambahan box ke 5.

## Versi 3. Konversi weight dari kilogram ke newton



Gambar 3.3 UML class diagram of vehicle demonstrating encapsulation





## Versi 3. Konversi weight dari kilogram ke newton

---

- Melakukan perhitungan yang mengubah satuan kapasitas box dari kilogram ke newton dengan menggunakan konsep enkapsulasi.



## Versi 3: Lakukan langkah-langkah berikut:

---

- Buatlah Vehicle2.java yang merupakan implementasi dari diagram UML pada gambar 3.3. (bisa kopi dari Vehicle1.java). Dan lakukan modifikasi sebagai berikut:
  - Modifikasi pada konstruktor dan pada method – method : getLoad dan getMaxLoad, dan addBox dengan tujuan untuk menambahkan fasilitas konversi dari kilogram (kilogram ini merupakan argumen pada konstruktor dan method-method tersebut) ke newton.



## Versi 3: Lakukan langkah-langkah berikut:

---

- Tambahkan method untuk konversi sebagai berikut:

```
private double kiloToNewts(double weight){  
    return (weight * 9.8);  
}  
private double newtsToKilo(double weight){  
    return (weight / 9.8);  
}
```

- Note: perhatikan bahwa secara eksternal data berat dalam satuan kilogram, tetapi secara internal terjadi konversi menjadi newton.
- Konversi ini terjadi pada saat pemanggilan method kiloToNewts() dan newtsToKilo().



## Versi 3: Ketik TestVehicle2.java berikut ini

```
public class TestVehicle {  
    public static void main(String[] args) {  
  
        // Create a vehicle that can handle 10,000 kilograms weight  
        System.out.println("Creating a vehicle with a 10,000kg maximum load.");  
        Vehicle vehicle = new Vehicle(10000.0);  
  
        // Add a few boxes  
        System.out.println("Add box #1 (500kg) : " + vehicle.addBox(500.0));  
        System.out.println("Add box #2 (250kg) : " + vehicle.addBox(250.0));  
        System.out.println("Add box #3 (5000kg) : " + vehicle.addBox(5000.0));  
        System.out.println("Add box #4 (4000kg) : " + vehicle.addBox(4000.0));  
        System.out.println("Add box #5 (300kg) : " + vehicle.addBox(300.0));  
  
        // Print out the final vehicle load  
        System.out.println("Vehicle load is " + vehicle.getLoad() + " kg");  
    }  
}
```

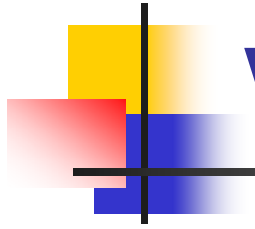


## Versi 3: tes TestVehicle2.java

---

- Kompile Vehicle2 dan TestVehicle2.
- Jalankan TestVehicle2, seharusnya keluar tampilan sebagai berikut:

```
Creating a vehicle with a 10,000kg maximum load.  
Add box #1 (500kg) : true  
Add box #2 (250kg) : true  
Add box #3 (5000kg) : true  
Add box #4 (4000kg) : true  
Add box #5 (300kg) : false  
Vehicle load is 9750.0 kg  
Finished executing  
,
```



## Versi 3: Resume

---

- Tampak pada output bahwa tidak ada perbedaan hasil dengan TestVehicle1.
- Hal ini terjadi karena konversi dilakukan secara tersembunyi → konsep enkapsulasi.
- Hal ini mengakibatkan kita tidak perlu merubah code pada TestVehicle2.java.