

Konstanta Final

1. Benahi kode Vehicle1 dan TestVehicle1 dan perbaiki jika menemui kesalahan!

Tidak ada kesalahan sintaks maupun logika pada kode tersebut. 

2. Hapus separator “/” pada file Vehicle1.java pada baris 4-6 serta pada file TestVehicle1.java pada baris 6, apa yang terjadi dan jelaskan! (Baris saya sesuaikan setelah kodennya dirapikan.)

A. Vehicle1

```
● ● ●
1 package KonstantaFinal;
2
3 public class Vehicle1 {
4     private double load;
5     private final double maxLoad = 10000;
6
7     public Vehicle1 (double max){
8         this.maxLoad = max; // Error karena variable maxLoad dideklarasikan sebagai final
9     }
}
```

Di sini, baris kedepalan mengalami error sintaks karena sebelumnya variabel `maxLoad` sudah dideklarasikan sebagai variabel bertipe **final** dan sudah memiliki nilai tetap yaitu `10000`. Oleh karena itu, nilai dari `maxLoad` tidak akan bisa diubah dalam kondisi **apapun**. Kita bisa menambahkan “/” kembali untuk menghindari error sintaks.

B. TestVehicle1

```
● ● ●
1 Vehicle1 vehicle2 = new Vehicle1(1000); // Error karena constructor Vehicle1 tidak ada yang menerima parameter
```

Di sini, baris tersebut akan mengalami error karena class `Vehicle1` tidak memiliki konstruktor yang menerima parameter `1000`. Constructor default tanpa parameter mungkin ada, tetapi untuk membuat objek dengan nilai maksimum ‘Load’ tertentu, diperlukan constructor tambahan yang menerima parameter `double`. Kita bisa menambahkan “/” kembali untuk menghindari error sintaks.

3. Pada file Vehicle1.java variabel `load` ubah menjadi konstanta final, apa yang terjadi, jelaskan!

```
1 package KonstantaFinal;
2
3 public class Vehicle1 {
4     private double final load;
5     private final double maxLoad = 10000;
6
7     // public Vehicle1 (double max){
8     //     this.maxLoad = max;
9     // }
10
11    public double getLoad() {
12        return this.load;
13    }
14
15    public double getMaxLoad() {
16        return this.maxLoad;
17    }
18
19    public boolean addBox(double weight) {
20        double temp = 0.0D;
21        temp = this.load + weight;
22        if (temp <= maxLoad) {
23            this.load = this.load + weight;
24            return true;
25        } else {
26            return false;
27        }
28    }
29 }
```

Di sini, kita akan melihat beberapa titik di mana error sintaks terjadi. Yang paling awal (baris empat) terjadi karena variabel `load` tidak diinisialisasi terlebih dahulu. Seharusnya, sebuah variabel bertipe `final` memiliki nilai **sejak awal** dan **tidak akan bisa diubah**. Error ini juga berimplikasi ke baris ke-12 karena baris tersebut tidak mengembalikan nilai apapun.

Lalu, sisanya terjadi karena variabel `this.load` berpotensi mengalami perubahan. Bisa dilihat bahwa di situ variabel `this.load` ditambahkan dengan `weight` yang nantinya akan mengubah nilai `this.load`. Ini mengalami error karena sebelumnya kita mendeklarasikan variabel `load` sebagai konstanta (`final`). Kita bisa kembali menghapus keyword “`final`” pada variabel `load` untuk menghindari error sintaks.

4. Tambahkan keyword “`static`” pada file `Vehicle1.java` variabel `maxLoad`, apa yang terjadi dan jelaskan!

```
● ● ●  
1 public class Vehicle1 {  
2     private double load;  
3     private static final double maxLoad = 10000;  
4  
5     ...  
6  
7 }
```



```
1 public double getMaxLoad() {  
2     return this.maxLoad;  
3 }  
4
```

Di sini, kita tidak mendapatkan error apapun. Namun, kita akan mendapatkan **satu warning**.

The static field Vehicle1.maxLoad should be accessed in a static way

Peringatan ini muncul karena kita menggunakan keyword “this” pada method `getMaxLoad()`. Keyword ini merujuk pada instance dari sebuah objek. Sedangkan, karena kita sudah menggunakan keyword “static” pada variabel `maxLoad`, variabel tersebut milik sebuah class dan nilainya akan sama / memengaruhi semua nilai `maxLoad` pada class tersebut.

Jika ingin menghilangkan peringatan tersebut, kita bisa menghilangkan keyword “this” pada variabel `maxLoad` di method `getMaxLoad()`.

```
● ● ●  
1 public double getMaxLoad() {  
2     return maxLoad;  
3 }
```