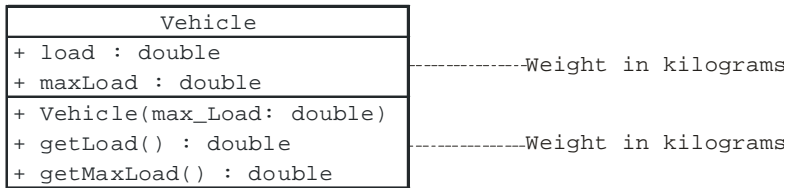


# PRAKTIKUM 1 PEMROGRAMAN BERORIENTASI OBYEK II

---

## Praktikum 1.1

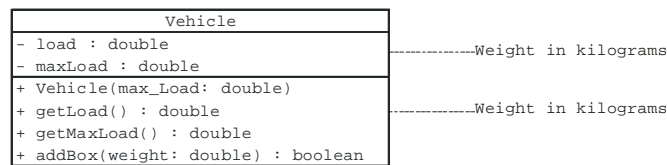


Dalam kelas `Vehicle` yang dideklarasikan di atas, semua atributnya memiliki mode akses public. Oleh karenanya, program `TestVehicle1` dapat mengakses atribut tersebut secara langsung.

1. Masuk ke dalam direktori kerja.
2. Buatlah sebuah kelas `Vehicle` yang mengimplementasikan diagram UML di atas.
  - ✓ Sertakan dua buah atribut publik, yaitu `load`: berat kendaraan pada saat sekarang dan `maxLoad`: batas maksimum beban kendaraan.
  - ✓ Sertakan sebuah konstruktor publik untuk mengeset nilai atribut `maxLoad`.
  - ✓ Sertakan dua buah method pengakses yang bersifat publik: `getLoad`, untuk mengembalikan nilai atribut `load` dan `getMaxLoad`, untuk mengembalikan nilai atribut `maxLoad`Semua data di atas dalam kilogram.
3. Bacalah kode program dalam `TestVehicle.java`. Program tersebut akan mengalami masalah ketika kotak yang terakhir ditambahkan ke dalam kendaraan, karena kode program tidak mengecek apakah penambahan kotak tersebut melebihi batas maksimumnya (`maxLoad`) atau tidak.
4. Jalankan kelas `TestVehicle`. Apa yang akan terjadi ?

## Praktikum 1.2

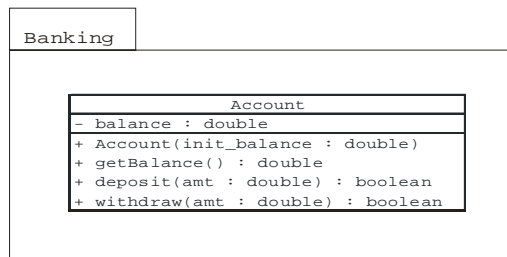
Untuk menyelesaikan masalah di atas, data atribut `load` dan `maxLoad` disembunyikan dan untuk mengaksesnya disediakan method `addBox`, yang akan melakukan proses pengecekan apakah beban kendaraan tersebut berlebih atau tidak.



1. Masuk ke dalam direktori kerja anda.
2. Salin kode program dari latihan sebelumnya, ubahlah kelas `Vehicle` tersebut sesuai dengan diagram UML di atas.
3. Bacalah kode program dalam `TestVehicle.java`. Program tersebut tidak dapat memodifikasi atribut `load` secara langsung, tetapi harus melalui method `addBox`. Method ini akan mengembalikan nilai `true` atau `false` dan akan ditampilkan di layar.
4. Kompilasi kelas `Vehicle` dan `TestVehicle`.
5. Jalankan kelas `TestVehicle`. ! Apa yang akan terjadi ?

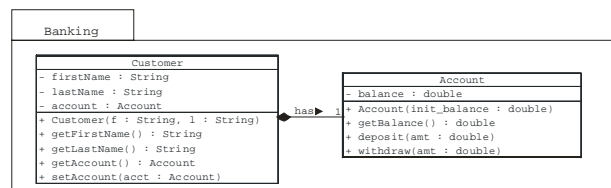
## Praktikum 1.3

Dalam latihan kali ini, anda diminta untuk membuat kelas `Account` sederhana. Source file nya ditempatkan ditempatkan di package `Banking`. Sebuah program untuk mengetes kelas ini yang bernama `TestBanking` telah dibuatkan di dalam package `default`, di mana di dalamnya ia membuat sebuah `account`. Kelas ini juga menginisialisasi nilai dari `account` tersebut dan melakukan beberapa transaksi sederhana, untuk kemudian mencetak nilai akhir dari `account` tersebut.



1. Buat sebuah direktori bernama banking.
2. Buat sebuah kelas `Account` di dalam file `Account.java` di dalam direktori `banking`. Kelas ini harus mengimplementasikan model dalam diagram UML di atas.
  - ✓ Deklarasikan sebuah atribut obyek yang bersifat `private`: `balance`. Atribut ini berisikan nilai `balance` dari `account` bank yang bersangkutan
  - ✓ Deklarasikan sebuah konstruktor publik yang menggunakan satu parameter (`init_balance`) yang mempopulasikan atribut `balance`
  - ✓ Deklarasikan sebuah method publik `getBalance` yang mengambil nilai `current balance`
  - ✓ Deklarasikan sebuah method publik `deposit` yang menambahkan nilai parameter `amount` ke dalam `current balance`
  - ✓ Deklarasikan sebuah method publik `withdraw` yang mengambil parameter `amount` dari nilai `current balance`
3. Di dalam direktori utama dari latihan ini, kompilasi *source file* `TestBanking.java`. Proses kompilasi ini juga akan mengkompilasi semua kelas yang digunakan dalam program, sehingga file `Account.java` akan terkompilasi juga di bawah direktori `banking`. `javac -d . TestBanking.java`
4. Jalankan program ! Apa yang akan terjadi ?

#### Praktikum 1.4



1. Buatlah sebuah direktori `banking`.
2. Salin file *project* `Banking` dari tugas sebelumnya ke dalam direktori ini.
3. Buatlah kelas `Customer` di dalam file `Customer.java` di dalam direktori `banking`. Direktori ini merepresentasikan struktur `package` program yang kita buat dalam bahasa Java. Kelas ini haruslah mengimplementasikan model diagram UML seperti di atas.
  - ✓ Deklarasikan tiga atribut yang bersifat `private`: `firstName`, `lastName`, dan `account`
  - ✓ Deklarasikan sebuah konstruktor publik yang mengambil dua parameter (`f` dan `l`) yang digunakan untuk mempopulasikan atribut obyek
  - ✓ Deklarasikan dua *accessor* yang bersifat publik untuk atribut obyek tersebut, method `getFirstName` dan `getLastName` yang mengembalikan atribut yang bersangkutan
  - ✓ Deklarasikan method `setAccount` untuk mengeset atribut `account`
  - ✓ Deklarasikan method `getAccount` untuk mengambil atribut `account`
4. Di dalam direktori utama, kompilasi dan jalankan program `TestBanking`. Apa yang akan terjadi ?

#### Praktikum 1.5

1. Buatlah sebuah aplikasi yang dinamakan dengan `TestArray`. Dalam method `main()`, deklarasikan dua buah variabel yaitu `array1` dan `array2`. Kedua variabel tersebut haruslah bertipe `int[]` (array dari integer).
2. Dengan menggunakan notasi kurung kurawal (`{}`), inialisasikan `array1` dengan nilai delapan bilangan prima pertama: 2,3,5,7,11,13,17 dan 19.
3. Tampilkan isi dari `array1`. Untuk hal tersebut dapat digunakan method `printArray` untuk menunjukkan nilainya. Kompilasikan program dan jalankan.
4. Berikan nilai variabel `array2` yang nilainya sama dengan `array1`. Modifikasi elemen yang berindex genap di dalam `array2` hingga nilainya sama dengan indeksinya (misal, `array2[0] = 0`, dan `array2[2] = 2`, dan seterusnya). Kompilasikan program `TestArray` tersebut dan jalankan. Perubahan apa yang terjadi pada array tersebut ?