

Concepts Orientés Objet

Leleux Laurent

2022 - 2023

Contexte

- UE : Programmation Java : Avancé
- COO / Ateliers Java
- Evaluation intégrée

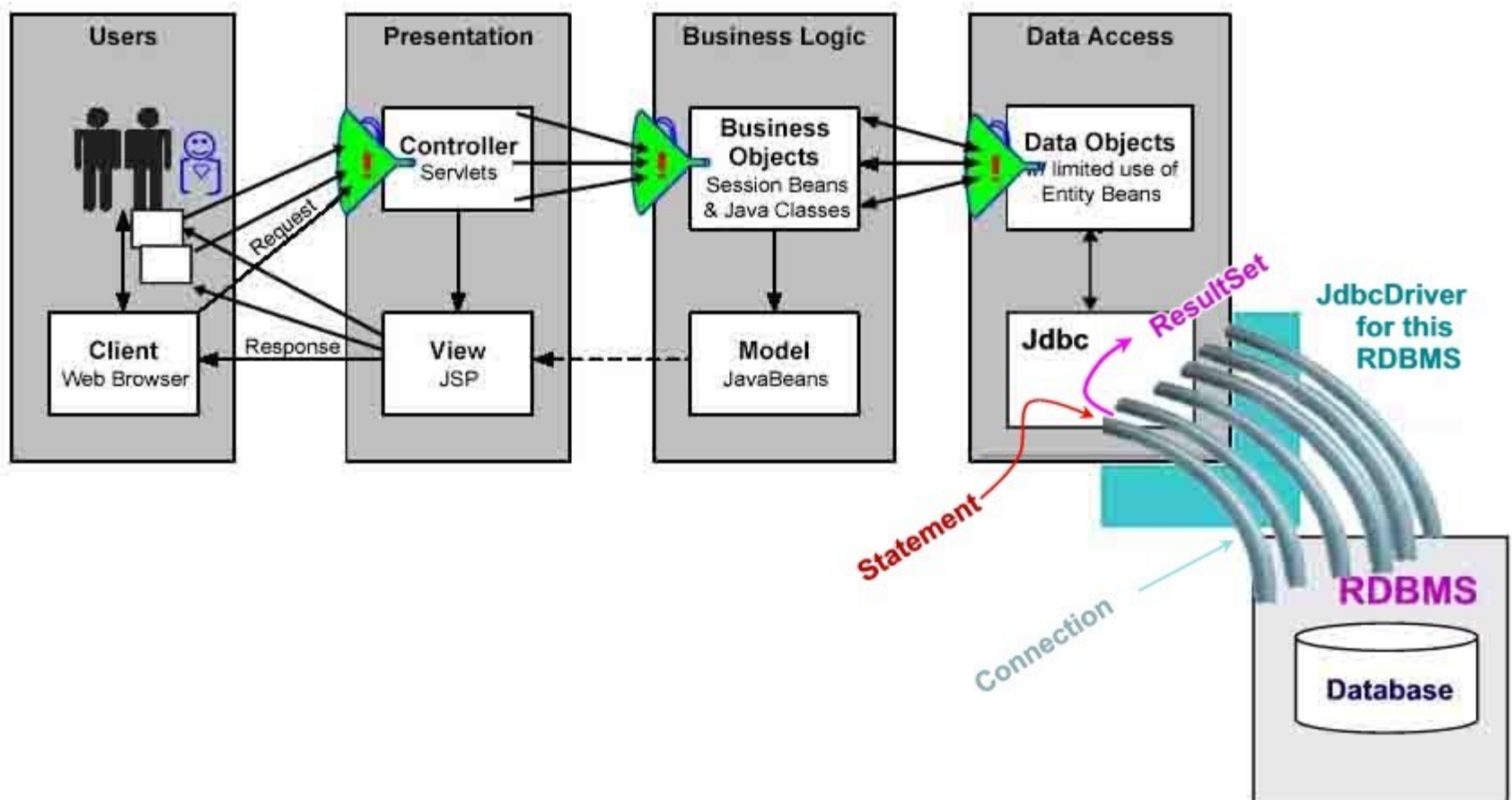
Objectifs

- Bonnes pratiques de design
- Orienté Objet avancé
- JAVA avancé
- Fonctionnement de la JVM
- Un framework d'applications multi-thread

Pourquoi ?

- Structure du code
- Communication
- Bugs

Application Multi-Thread



Vocabulaire

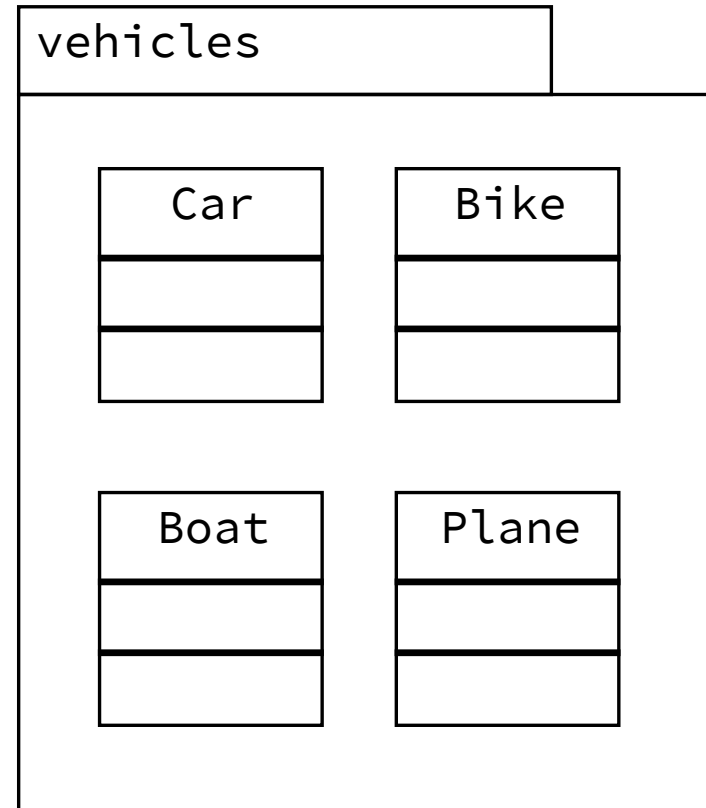
- Classe
- Instance
- Type
- Objet

Example

```
class Dog {  
    String barkSound = new String("Woof.");  
    boolean gentle = true;  
    int age = 5;  
}
```

Package

- Encapsulation
- Répertoire
- Namespace
- Librairie
- Importation

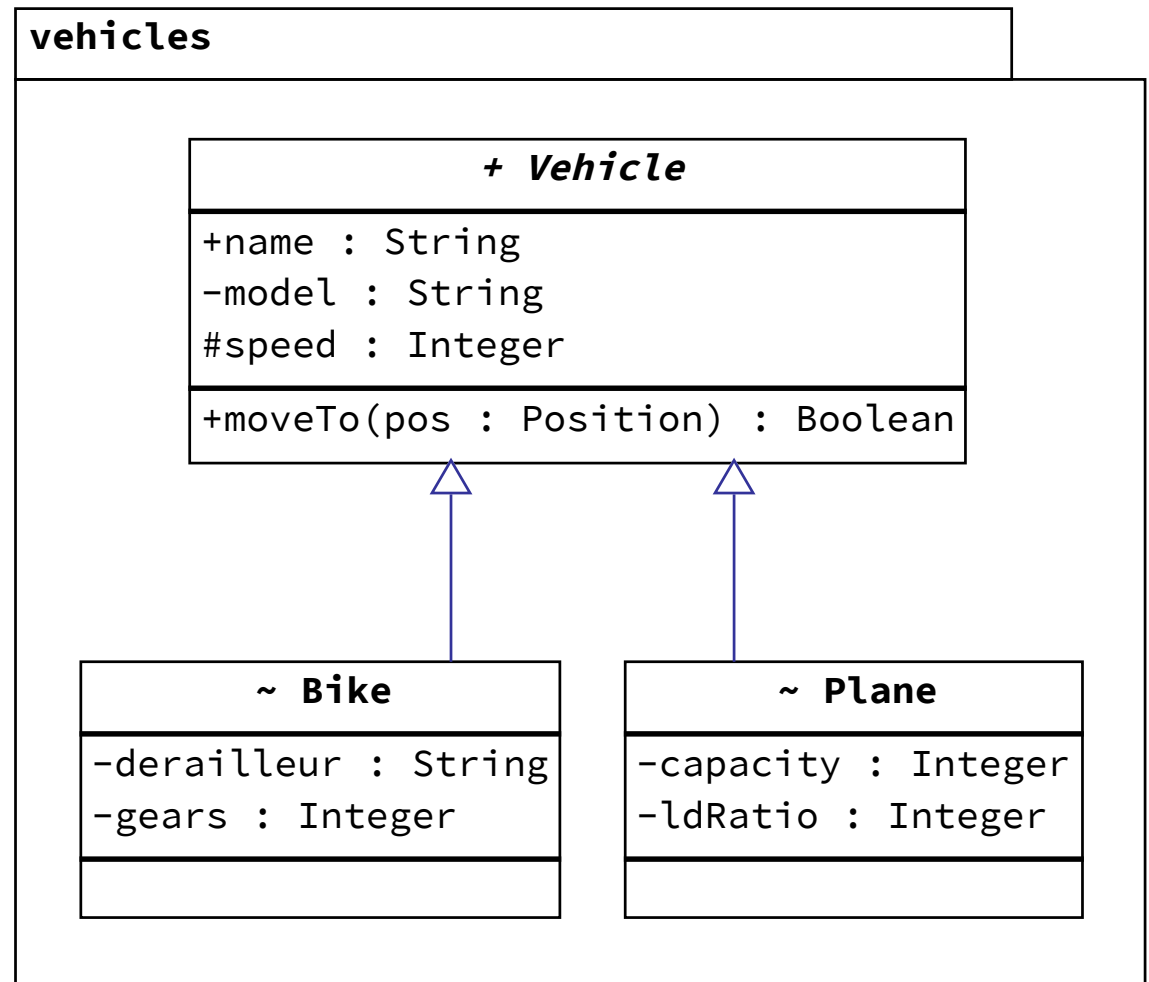


Package- Accessibilité

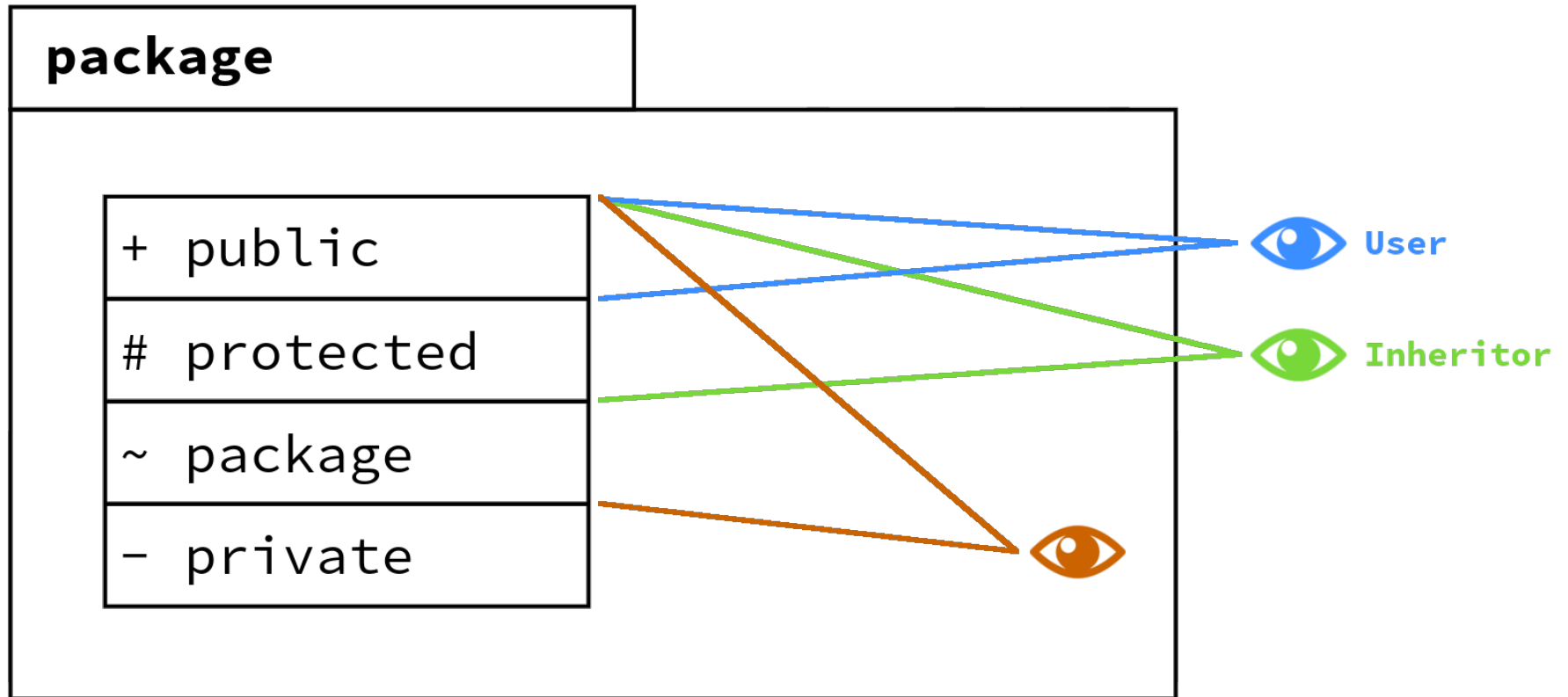
- Accès si :
 - Même package
 - Classe publique
- Plusieurs classes par fichier « .java »
- Une public
- Même nom

Un peu d'UML

- Package
- Classe
- Héritage
- Visibilité
(+ / - / # / ~)

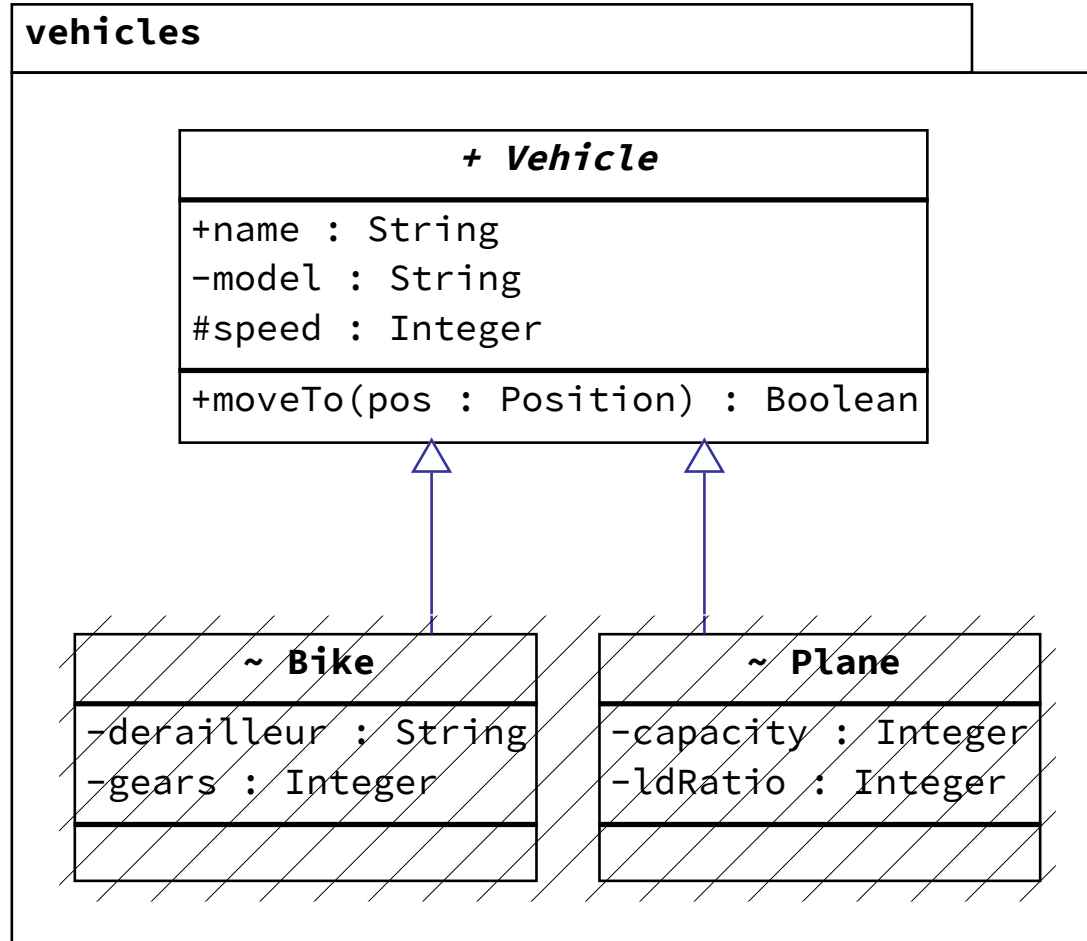


Visibilité



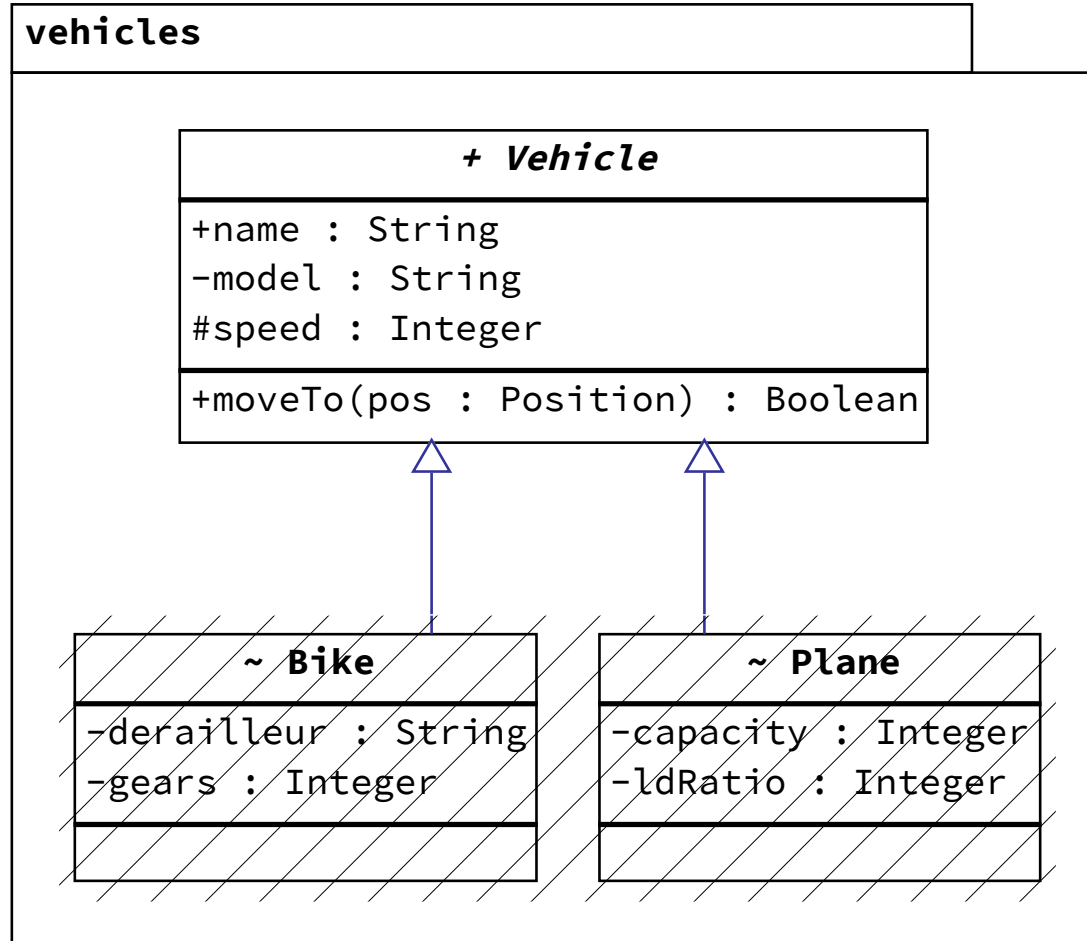
Exemples

Seules les classes « public » sont visibles en dehors du package.



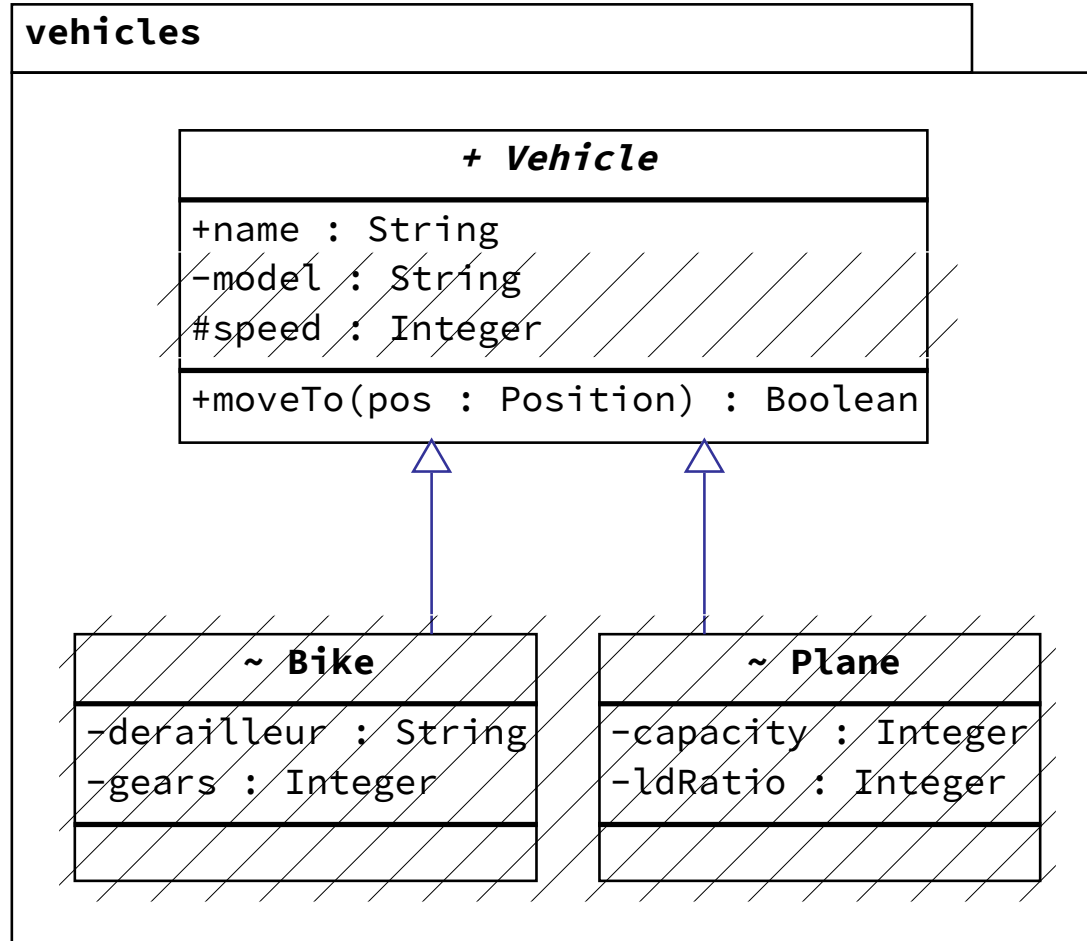
Exemples

De l'extérieur d'un package, impossible d'hériter d'une classe qui n'est pas « public ».



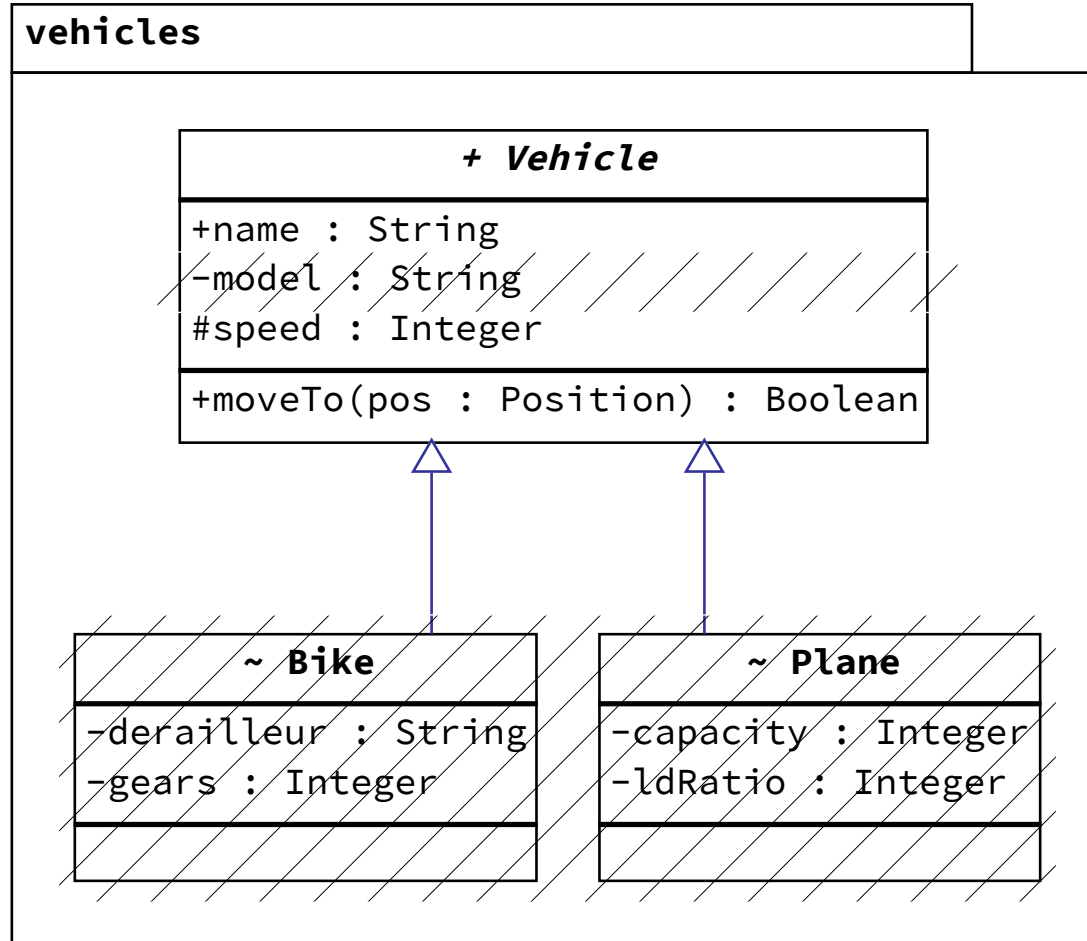
Exemples

De l'extérieur d'un package, on ne voit que les membres « public » des classes « public ».



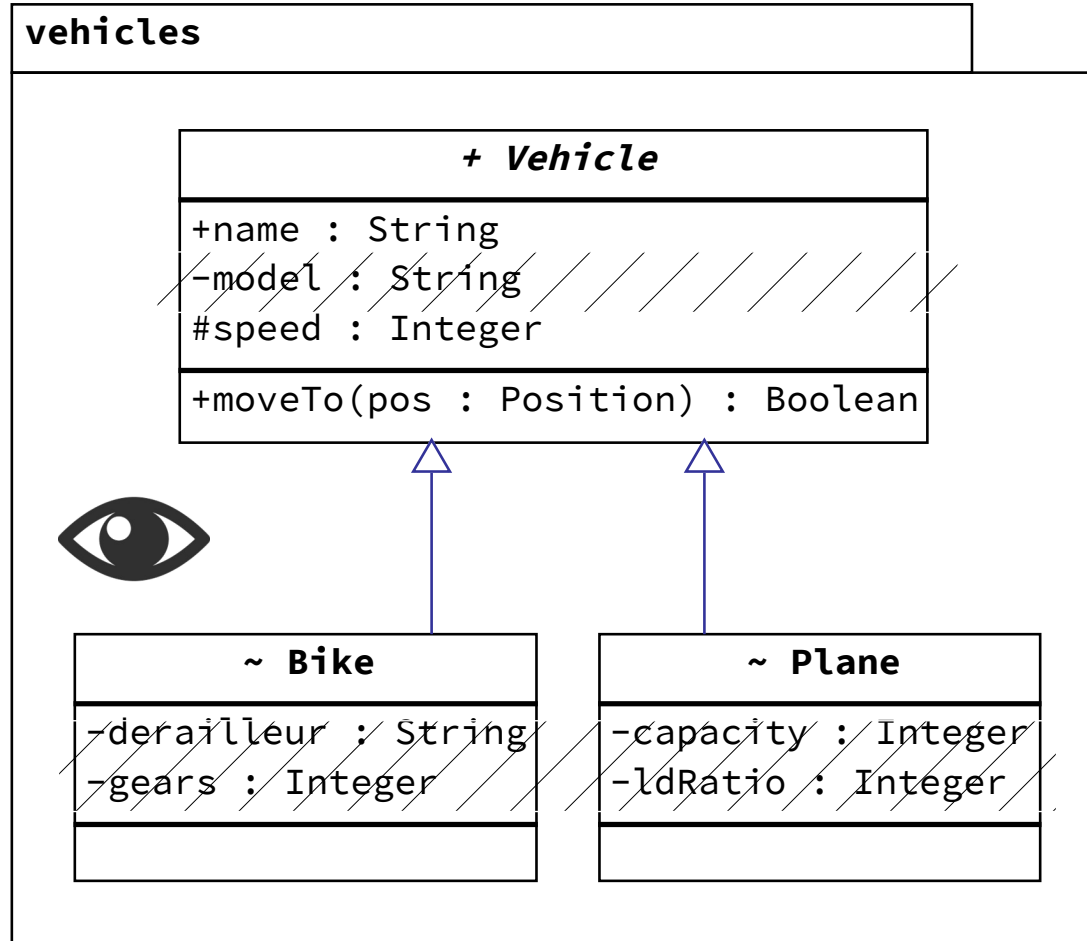
Exemples

De l'extérieur d'un package, les héritiers voient en plus les membres « protected » de sa super-classe.



Exemples

De l'intérieur d'un package, on voit tout sauf le « private » des autres classes.



Bonnes pratiques

« Encapsulez pour diminuer le couplage. »

- Le package ne montre que ce qui est utile aux voisins
- Réfléchissez lorsque vous mettez une classe « public ».