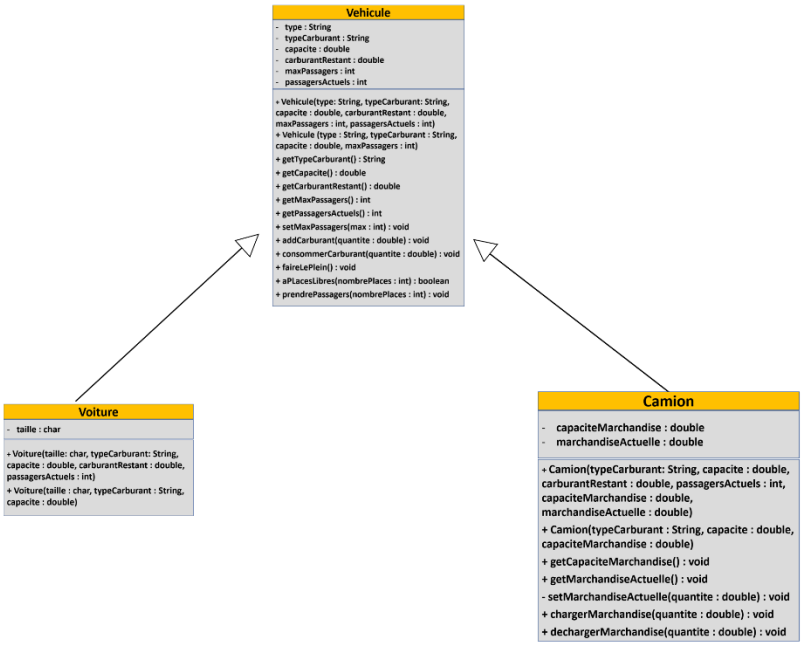
TD 4 L'héritage

Dans ce TD, nous allons utiliser le concept de l'héritage en tant que préparation pour notre prochain TP.

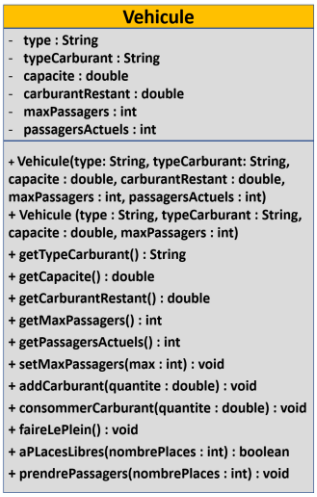
1. Vous aves les classes Java suivantes :
   Animal
   Arbre
   Humain
   Fleur
   Loup
   Lion
   Mammifère
   Insecte
   Pigeon
   Abeille
   Arbuste
   Plante
   - Ordonnez ces classes en indiquant quelle(s) classe(s) héritent de quelles classes.
   - Et si on rajoute la classe Organisme ?
   - Les objets dans toutes ces classes peuvent grandir. Par contre, seulement les animaux et les insectes peuvent bouger. Au contraire, seulement les plantes peuvent faire une photosynthèse pour se nourrir. Dans quelles classes ci-dessus voulez-vous ajouter les méthodes qui représentent ces trois capacités : grandir, bouger, photosynthèse ?

2. Nous allons simuler des divers types de véhicules : par exemple des voitures et des camions. Ces véhicules ont quelques caractéristiques communes, mais aussi des caractéristiques spéciales. Ils ont un type de carburant, une capacité du réservoir, un nombre maximal de passagers. Les deux types de véhicules peuvent faire le plein, consommer du carburant et prendre des passagers ; par contre, un camion pourra aussi charger/décharger de la marchandise.

   Ceci justifie l'utilisation de l'héritage. Voici un diagramme de classe de l'héritage concerné :



- Réalisez la classe **Vehicule**. Pour ce TP ce que nous intéresse en termes d'attributs, c'est le type de véhicule ("Voiture" ou "Camion"), le type de carburant qu'il utilise, la capacité de son réservoir, le volume de carburant dont il dispose maintenant, le nombre maximal de passagers y compris le chauffeur, ainsi que le nombre actuel de passagers. Il nous faudra un constructeur, des méthodes qui nous permettent de modifier des certains attributs (lesquels ?), des méthodes qui nous permettent de vérifier la valeur de certains attributs, une méthode **void consommerCarburant** (qui consume le combustible), une méthode **void faireLePlein** (qui remplira la voiture) et une méthode **void prendrePassager**.



Consultez les diagrammes de classe ci-dessus.

- Identifiez les constructeurs
- Ecrivez le premier constructeur
- Pour le deuxième constructeur, le carburant restant sera mis par défaut à la valeur de la capacité du réservoir, tandis que le nombre actuel de passagers sera 1 (le chauffeur). Ecrivez ce constructeur
- La classe **Vehicule** contient plusieurs méthodes get, qui retournent chacune la valeur d'un attribute de la classe. Identifiez-les dans le diagramme de classe.
- Ecrivez la méthode double getCapacite().
- La méthode **setMaxPassagers(int max)** nous permet de choisir le nombre maximal de passagers qu'on peut avoir dans un véhicule, y compris le chauffeur. Ecrivez cette méthode
- Les méthodes **addCarburant(double quantite)** et **consommerCarburant(double quantite)** nous permettent de modifier la quantité de carburant actuelle qui existe dans la voiture. Ecrivez la méthode **addCarburant.**
- Selon vous, quel est le fonctionnement des dernières trois méthodes ? Ecrivez-les !

- Réalisez la classe **Voiture** qui hérite de la classe **Vehicule** (notamment les objets de la classe voiture mettront l'attribut type de la classe Voiture à "Voiture"). En plus des attributs hérités de sa superclasse, la sous-classe **Voiture** aura un attribut taille, qui sera de type char et qui fera la différence entre une petite voiture -- avec 2 passagers max, une de taille moyenne -- max 5 passagers et une de grande taille -- max 8 passagers. Faites un constructeur pour la classe **Voiture** en utilisant celui de la classe **Vehicule**.

- Pour la classe **Camion** on limite le nombre de passagers à 2 dans la cabine. Nous aurons besoin de 2 attributs de type double en plus de ceux de la classe **Vehicule**, qui mesurent le poids maximal que le camion peut porter, ainsi que la quantité actuelle de marchandise qu'il porte. La classe **Camion** aura des méthodes pour charger une certaine quantité de marchandise, et aussi pour la décharger. Les méthodes pourraient s'appeler void chargerMarchandise (double quantite) et void dechargerMarchandise (double quantite). Regardez le diagramme de classe ci-dessus.

  Ecrivez les deux constructeurs de la classe Camion, ainsi que les méthodes chargerMarchandise(double quantite) et dechargerMarchandise(double quantite)

- Disons que dans une classe principale nous avons le code suivant :

```
public static void main(String[] args){
        Vehicule[] vehicules = new Vehicule[3];
}
```

Les 5 éléments de ce tableau seront, en ordre : une voiture de taille moyenne, une voiture de taille petite, un camion, et une voiture de grande taille. Vous pouvez choisir librement les paramètres à mettre dans les constructeurs, sauf la carburant actuel, qui est mis à ¼ de la capacité maximale.
Ecrivez du code pour initialiser les éléments du tableau.

- Ce code, c'est un exemple de quel concept de Programmation Orientée Objet ?
- Pourquoi est-ce qu'on peut mettre un objet, disons de type Camion, dans un tableau du type **Vehicules[]** ?
- Disons qu'on veut que chaque véhicule fasse le plein. Écrivez du code dans la classe principale qui peut faire cela.
- On suppose qu'il y a deux passagers dans chaque vehicule. Ecrivez du code qui essaie de faire chaque vehicule prendre un nouveau passager. Quel devrait être le résultat ?
- Qu'est-ce que se passe lorsqu'on écrit la ligne de code dans la classe principale :

```
vehicule[3].chargerMarchandise(120) ; ?
```