

TP-4 (Java POO)

Héritage et Polymorphisme

1. Définissez une classe **Vehicule** qui a pour attributs des informations valables pour tout type de véhicule : sa marque, son année d'achat, son prix d'achat et son prix courant. Vous devez vous assurer de la bonne encapsulation de ces attributs.
2. Définissez un constructeur prenant en paramètres la marque, la date d'achat et le prix d'achat.
3. Définissez une méthode publique `afficher()` qui affiche la valeur des attributs.
4. Définissez deux classes **Voiture** et **Avion**, héritant de la classe **Vehicule** et ayant les attributs supplémentaires suivants :

- Pour la classe Voiture :

`cylindree, nbPortes, puissance et kilometrage`

- Pour la classe Avion :

`moteur et nbHeuresVol`

5. Définissez maintenant un constructeur pour **Voiture**, ainsi qu'une méthode `afficher()` qui affiche la valeur des attributs.

Ces deux méthodes doivent, bien entendu, être publiques puisqu'elles sont précisément faites pour être utilisées hors de la classe.

Notez que pour le constructeur de **Voiture**, on fait appel au constructeur de **Vehicule** (classe mère). On fait également appel à la méthode d'affichage de la super-classe dans la méthode `afficher()` de **Voiture**.

Définissez maintenant le Constructeur et la méthode `afficher()` de la classe **Avion** qui s'implémentent de la même façon que pour **Voiture**.

6. Ajoutez dans la classe **Vehicule** une méthode `calculerPrix()` qui permet de calculer et d'afficher le prix courant. Cette méthode ne retourne rien.

Le prix doit rester positif (donc s'il est négatif, on le met à 0).

Règle de calcul :

=> $\text{decote} = (\text{annee Actuelle} - \text{date d'Achat}) * 0,01$

=> $\text{prix courant} = (1 - \text{decote}) * \text{prixAchat}$

Redéfinissez cette méthode dans les deux sous-classes **Voiture** et **Avion**.

Les signatures de cette méthode sont les mêmes pour **Voiture** et **Avion** que pour **Vehicule**, par contre les définitions (les comportements) diffèrent.

Règle de calcul pour un Avion :

=>

- si le moteur est à Hélices :

alors $\text{decote} = 1 \text{ millième des heures de Vol}$

- si autre type de moteur :

alors $\text{decote} = 1/10000 \text{ des heures de vol}$

=>

Prix Courant de l'avion = $(1 - \text{decote}) * \text{Prix d'Achat}$

Règle de calcul pour une voiture :

```

=>
decote = (annee actuelle - date d'achat) * 0,02
decote = decote + (kilométrage / 10000) * 0,05
ensuite :
si voiture de marque FIAT ou RENAULT :
decote = decote + 0,1
si voiture de marque FERRARI ou PORSCHE :
decote = decote - 0,2
=>
prix courant = (1,0 - decote) * prix d'achat

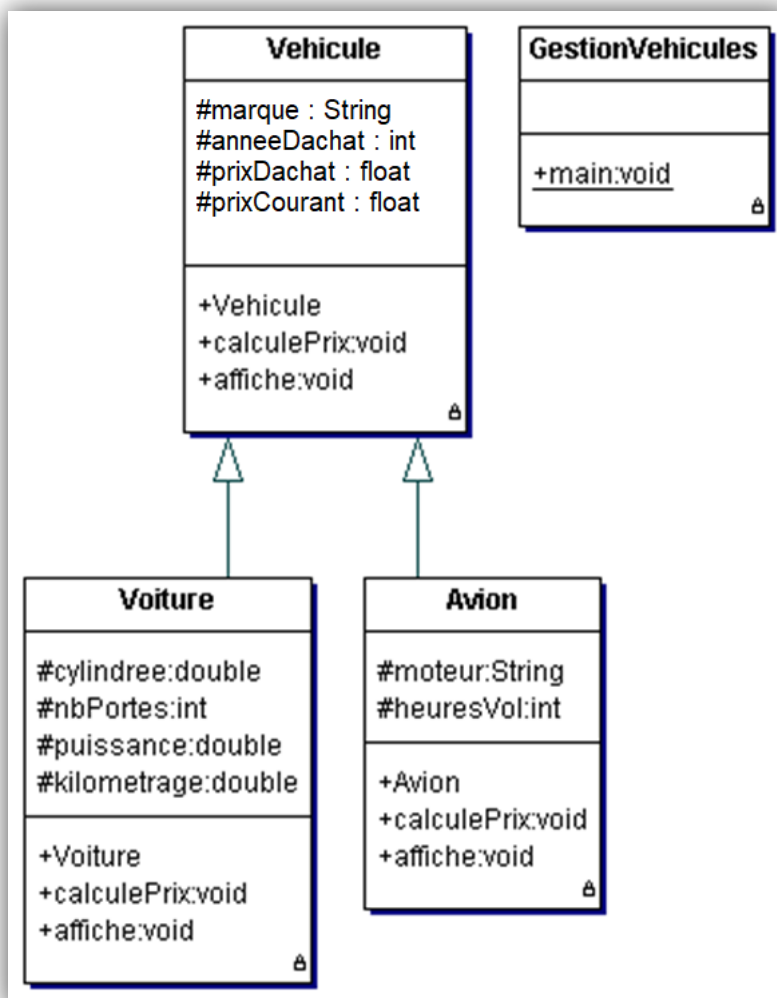
```

A ce niveau là du TP, vous allez vous rendre compte ici que l'on a besoin d'accéder à des attributs privés de la super-classe, ou de les modifier (*anneeAchat*, *marque*, *prixAchat* et *prixCourant*). Il faut donc enrichir la classe **Vehicule** des "getters/setters" nécessaires

Il aurait été possible de déclarer les attributs nécessaires comme **protected** dans la classe **Vehicule** pour s'éviter la peine de définir les getters/setters. Ceci peut cependant nuire à une bonne encapsulation : un autre programmeur peut hériter de votre classe **Vehicule**, il aurait alors accès aux détails d'implémentation et vous ne pourriez plus modifier librement cette implémentation sans potentiellement causer du tort à ses programmes !

Voici à quoi ressemble le schéma UML final :

NB : dans ce schéma, remplacez le paramètre *dateAchat* par *anneeAchat* (de type **int**).



Ecrivez le code de la classe qui contient le code main ci-après :

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

    List<Vehicule> vehicules = new ArrayList<Vehicule>();

    Voiture v1 = new Voiture(2.5f, 5, 120, 120000,
                             MarqueVehicule.PEUGEOT.toString(), 2005, 5000.0f);

    Voiture v2 = new Voiture(6.5f, 2, 280, 81300,
                             MarqueVehicule.PORSCHE.toString(), 1999, 250000);

    Avion a1 = new Avion(250, TypeMoteur.HELICES.toString(),
                         MarqueVehicule.AVION_CESSNA.toString(), 1982, 1230673.90f);

    Avion a2 = new Avion(1300, TypeMoteur.REACTION.toString(),
                         MarqueVehicule.AVION_NAIN_CONNU.toString(), 1993, 4321098.00f);

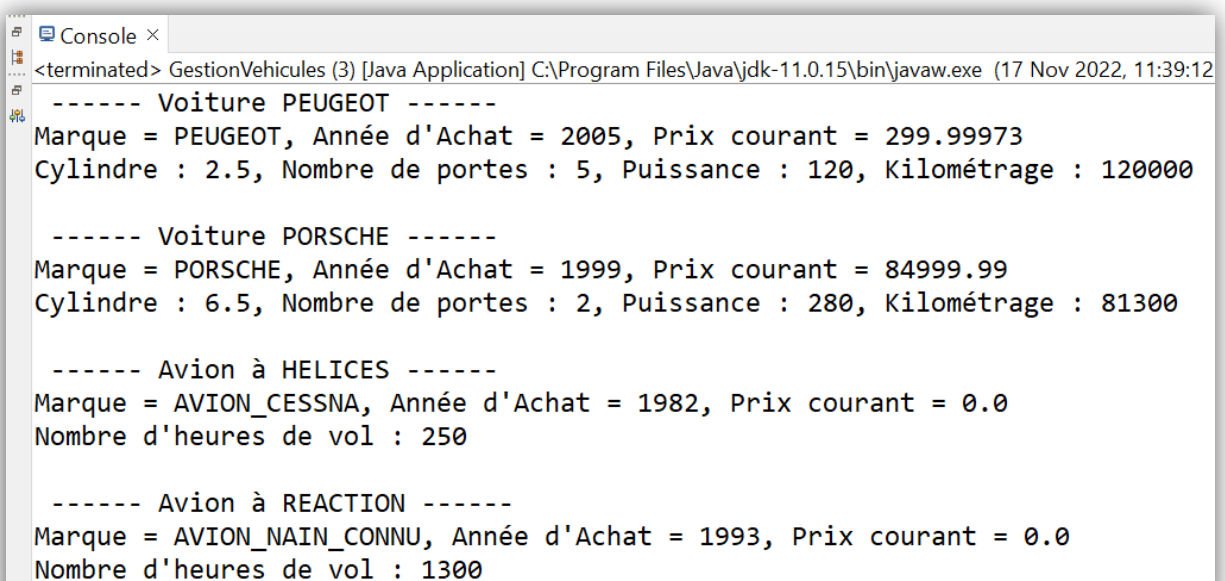
    vehicules.add(v1);
    vehicules.add(v2);
    vehicules.add(a1);
    vehicules.add(a2);

    Calendar calendar = Calendar.getInstance();

    for (Vehicule v : vehicules) {
        v.calculerPrix(calendar.get(Calendar.YEAR));
        v.afficher();
        System.out.println();
    }
}
```

Vous pouvez remplacer la boucle **for** par la boucle **foreach**.

Exécutez le code. Vous devez obtenir le résultat suivant :



```
<terminated> GestionVehicules (3) [Java Application] C:\Program Files\Java\jdk-11.0.15\bin\javaw.exe (17 Nov 2022, 11:39:12)

----- Voiture PEUGEOT -----
Marque = PEUGEOT, Année d'Achat = 2005, Prix courant = 299.99973
Cylindre : 2.5, Nombre de portes : 5, Puissance : 120, Kilométrage : 120000

----- Voiture PORSCHE -----
Marque = PORSCHE, Année d'Achat = 1999, Prix courant = 84999.99
Cylindre : 6.5, Nombre de portes : 2, Puissance : 280, Kilométrage : 81300

----- Avion à HELICES -----
Marque = AVION_CESSNA, Année d'Achat = 1982, Prix courant = 0.0
Nombre d'heures de vol : 250

----- Avion à REACTION -----
Marque = AVION_NAIN_CONNU, Année d'Achat = 1993, Prix courant = 0.0
Nombre d'heures de vol : 1300
```