P.O.O. : Java — Sujet n°2 Bis
Révisions sur les notions d'héritage et d'interfaces

Pensez à créer un **nouveau projet** (`Sujet2Bis`) puis un **package** qui contiendra toutes les classes ainsi que l'interface nécessaires à l'exercice ci-dessous.

La classe `Vehicule`

Un collectionneur veut gérer la liste des véhicules qu'il possède. Il collectionne différents types de véhicules : des voitures, des motos et des vélos. Chaque véhicule possède

- un *modèle* (exemple : Triumph Boneville),
- une *valeur* marchande (exemple : 9390,50€),
- une *cylindrée* (exemple : 865cm^3).

Il désire gérer la liste de ses véhicules, savoir combien de véhicules il possède ...

Écrire la classe `Vehicule` avec :

1. les *variables d'instances* et la *variable de classe* nécessaires,
2. les constructeurs qui permettent d'« enregistrer » (et de comptabiliser) tous les véhicules en spécifiant leur *marque*, leur *valeur* marchande et leur *cylindrée.* Bien entendu dans le cas des vélos, la cylindrée n'existe pas … elle sera donc initialisée à 0,
3. la methode `getValeur` qui retourne la valeur marchande,
4. la redéfinition de la méthode `toString` pour qu'elle produise une chaîne spécifiant le modèle, la valeur marchande et la cylindrée. Si l'on reprend l'exemple ci-dessus, la chaîne produite sera : « Triumph Bonneville (865cm^3) : 9390,50€ ». Dans le cas où la cylindrée n'existe pas, elle sera remplacée par la mention : « véhicule non motorisé »,
5. la méthode, `cptVehicules`, qui affiche simplement combien de véhicules ont été enregistrés,
6. la méthode, `plusChere`, qui comparera le *véhicule courant* avec un *autre véhicule passé en paramètre* et qui *retournera* celui ayant la plus grande valeur marchande.

Écrire la classe `Principal` contenant uniquement la méthode `main` qui va permettre de tester les deux constructeurs et les méthodes `cptVehicules`, `toString` et `plusChere`.

L'interface `Deplacement`

Tous les véhicules accélèrent et freinent … mais ils ne le font pas tous de la même façon (on n'accélère pas de la même façon avec un vélo ou avec une moto!).

Écrire l'interface `Deplacement` qui contiendra les méthodes abstraite `accelerer` et `freiner` qui, comme la méthode `toString`, retourne une chaîne de caractères décrivant l'action.

Les classes `Voiture, Moto et Velo`

Les voitures, motos et vélos sont des véhicules :

- les voitures ont en plus un *nombre de places*, leur *accélération* se fait au pied avec la pédale de droite, pour *freiner* c'est la pédale du milieu,

- les motos ont en plus un *type de selle* (solo, duo, en cuir, …), leur *accélération* se fait à la main en utilisant la poignée des gaz, pour *freiner* c'est avec le levier de freins,
- les vélos n'ont aucune caractéristiques supplémentaires, pour *accélérer* il faut simplement pédaler plus fort, pour *freiner* on peut user ses semelles ;-)
- les 3 classes redéfiniront la méthode `toString` en indiquant de quel véhicule il s'agit (« Voiture », « Moto » ou « Velo ») suivi des caractéristiques communes à tous les véhicules puis des caractéristiques spécifiques (nombre de places pour les voitures, le type de selle pour les motos).

Écrire les classes `Moto`, `Voiture` et `Velo` avec leur constructeur, leur redéfinition de la méthode `toString`, et l'implémentation des méthodes abstraites `accelerer` et `freiner`.

Modifier la classe `Principal` afin de :
- créer une moto, une voiture et un vélo,
- déclarer un tableau de véhicules (utiliser la *variable de classe* contenant le nombre de véhicules construits pour donner sa *taille*), et y stocker tous les véhicules créés aux questions précédentes,
- calculer puis afficher la valeur marchande de la totalité de la collection.