Programmation Objet en Java - Feuille de TD 8

Exercice 1. Dans cet exercice, nous allons modéliser des animaux en s'intéressant particulièrement à leur vitesse de déplacement (sur terre).

1. Un animal est défini par sa taille (entière) en cm et sa vitesse en km/h, entière également. Définir un objet `Animal` avec ses deux champs `size` et `speed` et de quoi le construire. Doit-on écrire un record ou une classe ?

2. Un animal est rapide si sa vitesse est supérieure à 10 fois sa taille. Ajouter une méthode `isFast` permettant de tester si un animal est rapide.

3. Ajouter ce qu'il faut pour pouvoir afficher un animal comme suit :

```
var pangolin = new Animal(30,1);
System.out.println(pangolin); // size = 30 cm, speed = 1km/h
```

4. Les chevaux sont des animaux un peu particuliers : ils sont spécialisés pour la course et leur qualité dépend du nombre d'années qu'ils ont passées à courir. Écrire la classe `Horse` qui possède les même caractéristiques qu'un `Animal` avec un champs supplémentaire `runningYears` indiquant le nombre d'années passées à courir.

5. Ajouter ce qu'il faut pour pouvoir afficher un cheval comme suit :

```
var jollyumper = new Horse(210,20,8);
System.out.println(jollyjumper);
// Horse : running years = 8 years, size = 210 cm, speed = 20km/h
```

6. Peut-on utiliser la méthode `isFast` sur un cheval ?

7. Un cheval a deux allures de déplacement, l'allure normale, qui correspond à sa vitesse et la course. Sa vitesse de course est égale à sa vitesse normale, multipliée par 30 et divisée par le nombre d'années passées à courir. Écrire une méthode `runningSpeed` qui donne la vitesse de course d'un cheval.

8. Les licornes (de course) sont comme les chevaux, sauf qu'elles ont une corne en plus. Écrire la classe `Unicorn` représentant des licornes.

9. La corne est un handicap pour courir : les licornes courent 2 fois moins vite que les chevaux ayant les mêmes caractéristiques. Faire en sorte que la méthode `runningSpeed` renvoie la bonne valeur pour une licorne:

```
var lily = new Unicorn(150, 40, 3);
System.out.println(lily.runningSpeed()); // 200
```

10. Ajouter (et modifier) ce qu'il faut pour pouvoir afficher une licorne ainsi:

```
System.out.println(lily);
// Unicorn : running years = 3 years, size = 150 cm, speed = 40km/h
```

Exercice 2. On souhaite rajouter des manchots.

1. Les manchots sont des animaux qui peuvent se déplacer normalement (vitesse de déplacement donnée par `speed()`), mais aussi dans l'eau. Par rapport aux autres animaux, ils ont donc une vitesse de déplacement sous l'eau (`underWaterSpeed`) en plus. Ils ont aussi une vitesse de déplacement moyenne donnée par `averageSpeed()` qui est la moyenne de ces deux vitesses. Écrire la classe `Penguin`, avec les différentes méthodes renvoyant une vitesse.

Exercice 3. Proposer une autre architecture pour implémenter `Animal`, `Horse`, `Unicorn`, `Penguin` qui ne comporte aucune classe dérivée. On créera une interface `Animal`, une sous-interface `Running` de `Animal` pour les animaux qui ont une méthode `runningSpeed()` et une sous-interface `Marine` de `Animal` pour les animaux marins qui ont une méthode `underwaterSpeed()` et une méthode `averageSpeed()`. Les classes `Horse`, `Unicorn` seront maintenant des record implémentant `Running` et `Penguin` sera un record implémentant `Running`.