Laboratoire 04 (partie 2) – Les classes et les objets

OBJECTIFS :

- Approfondir votre compréhension de la définition et l'utilisation de classes (attributs d'instance et de classe, constructeurs, getters, setters, méthodes d'instance et de classe…).
- Continuer à vous exercer à coder en Java, en respectant les bonnes pratiques, et les normes de programmation (style Java).

Votre travail consiste à définir une classe `Animal` qui modélise un animal de compagnie.

Prenez soin de bien documenter votre code et n'oubliez pas les commentaires de documentation (Javadoc) sur l'entête de chacune de vos méthodes.

ATTENTION : Si vous voulez utiliser la classe `TestsAnimal.java` (qui vous est fournie, et qui sert à tester partiellement la classe `Animal.java` que vous devez faire), utilisez les mêmes noms de méthodes, et le même ordre des paramètres formels dans vos définitions. Sinon, vous devrez modifier les noms et l'ordre des paramètres effectifs des appels dans le `main` de la classe `TestsAnimal`.

EXERCICE 1

1. Définition des constantes de classe : vous devez mettre ces 2 constantes de classe dans votre classe, et les utiliser au possible.

```
//nom des sortes, selon leur code
public final static int SORTE_0 = 0;
public final static int SORTE_1 = 1;
public final static int SORTE_2 = 2;
public final static int SORTE_3 = 3;

//pour obtenir le nom d'une sorte avec son code
//par exemple, le nom de la sorte = SORTE_0 est TAB_SORTES[sorte] => "chat".
public final static String [] TAB_SORTES
                                = {"chat", "chien", "oiseau", "autre"};
```

1. Définitions des attributs d'instance : un animal est défini avec 4 attributs :
   a. nom (String) // le nom de l'animal
   b. sorte (int) // SORTE_0, SORTE_1, SORTE_2 ou SORTE_3
   c. proprietaire (String) // le nom du propriétaire de l'animal
   d. avecMicropuce (boolean) // true si l'animal a une micropuce, false sinon.

2. Définition des constructeurs et méthodes d'instance :

   NOTES :

   ANTÉCÉDENT pour toutes les méthodes (ou constructeurs) qui reçoivent en paramètre(s) le nom de l'animal et / ou celui du propriétaire : ces paramètres ne doivent pas être `null`.

   ANTÉCÉDENT pour toutes les méthodes (ou constructeurs) qui reçoivent en paramètre(s) la sorte d'animal : ce paramètre doit être une valeur parmi : SORTE_0, SORTE_1, SORTE_2 ou SORTE_3

   RÉUTILISEZ le code autant que possible (attention à la réutilisation de constructeur !)

a. Codez 3 constructeurs :
   i. Un constructeur d'initialisation qui prend en paramètre des valeurs pour initialiser tous les attributs de l'animal à construire (voir la classe de tests pour l'ordre des paramètres).
   ii. Un constructeur de copie qui prend en paramètre un autre animal, et qui en fait une copie (contenus identiques, mais références différentes).
   iii. Un constructeur sans argument qui initialise les attributs ainsi : le nom de l'animal et le nom du propriétaire avec la chaine vide, la sorte à SORTE_0, et sans micropuce.

b. Codez les *getters* et les *setters* pour tous les attributs.

c. Codez la méthode `toString` (sans paramètre) qui retourne une représentation sous forme de chaine de caractères de cet animal (voir la classe de tests pour le format). NOTE : utilisez TAB_SORTES pour obtenir le nom de la sorte.

   Exemple du format qui doit être affiché avec micropuce :

```
Nom     : Ratatouille(avec micropuce)
Sorte   : autre
Proprio : France Dufour
```

   Ou sans micropuce :

```
Nom     : Titi(sans micropuce)
Sorte   : oiseau
Proprio : Mimi Gonzalez
```

d. Codez la méthode `equals` qui prend en paramètre un animal, et qui retourne si oui ou non (`boolean`) le proprietaire de cet animal est le même que celui de l'animal donné en paramètre.

   Rappel pour comparer des chaines de caractères : `s1.equals(s2)`. Ou si l'on ne veut pas tenir compte de la casse : `s1.equalsIgnoreCase(s2)`.

3. Définitions d'attributs et de méthodes de classe :

a. Modifier votre classe pour qu'elle possède un attribut de classe `nbrAnimauxAvecMicropuce` permettant de tenir le compte de tous les animaux créés qui ont une micropuce. Cet attribut est initialisé à 0. N'OUBLIEZ pas de modifier le code de vos constructeurs et méthodes concernées pour ajuster cette valeur lorsqu'il y a lieu.

   NOTE : lorsqu'on fait une copie d'un animal qui possède une micropuce (avec le constructeur de copie), il compte pour un animal de plus ayant une micropuce.

b. Ajoutez la méthode de classe `getNbrAnimauxAvecMicropuce`, sans paramètre, qui retourne la valeur de `nbrAnimauxAvecMicropuce`.

c. Ajoutez la méthode de classe `decrementerNbrAnimauxAvecMicropuce`, qui ne retourne rien et qui prend en paramètre le nombre qu'on doit soustraire au `nbrAnimauxAvecMicropuce`.

d. Testez votre classe avec la classe de tests fournie, et AJOUTEZ-Y vos propres tests : créez des objets, et appelez les méthodes d'instance sur ces objets. Testez aussi vos méthodes de classe en les appelant sur le nom de la classe (par sur un objet).

BON TRAVAIL !