université PARIS DIDEROT PARIS 7

TYPES DE DONNÉES ET OBJETS (TO2)

TD1 : Objets, attributs et constructeurs

1 Des Animaux

On veut définir une classe `Animal`. Elle aura la liste des attributs publics suivante :
- un attribut `nom` qui sera une chaîne de caractères,
- un attribut `genre` qui vaudra `'m'` pour un mâle, `'f'` pour une femelle (ou indéterminé).
- un attribut `poids` qui sera un entier donnant le poids de l'animal en grammes,
- un attribut `espece` qui sera une chaîne de caractères,
- un attribut `age` qui sera un entier donnant l'âge de l'animal en jours.

1. Écrire une classe `Animal` répondant à ce cahier des charges.
2. Que doit-on écrire pour créer un zèbre mâle de 5 ans, pesant 300 kg s'appelant Marti ? un hippopotame femelle de 7 ans, d'une tonne et demie s'appelant Gloria ?
3. Écrire un constructeur permettant d'initialiser les attributs au moment de l'initialisation de l'objet.
4. Que doit-on écrire pour diminuer le poids de Gloria de 50 kg ?
5. Que penser du code suivant ?

```
Animal a = new Animal("Melman",'m',4,1000,"girafe");
Animal b = a;
b.poids = 950;
System.out.println(a.poids);
```

6. Écrire une méthode `description` qui prend un `Animal` et qui renvoie une chaîne de caractères le décrivant (”Je m'appelle Rico, je suis un manchot male, j'ai 3 ans et je pese 30 kg”).
7. On ajoute un attribut entier `faim` à cette classe qui représente la degré de faim de l'animal et qui varie entre `0` et `10`. S'il vaut `0` l'animal n'a pas faim, s'il est à `10`, il est très affamé. Au début un animal sera à `5`.

   Écrire une méthode `nourrir` qui diminue la faim de `5`.
8. Écrire une méthode statique `plusGros` qui prend en argument un tableau d'animaux et qui renvoie un élément de poids maximal.
9. Écrire un `main` dans lequel vous initialiserez un tableau avec trois animaux pour lui appliquer la méthode `plusGros` précédente.
10. Écrire une méthode statique `reproduction`, prenant en argument deux animaux et qui, s'ils sont de sexes opposés et de la même espèce, et si leurs niveaux de faim sont inférieurs à 5, renvoie un nouvel animal de la même espèce et de sexe tiré aléatoirement. Le poids sera un nombre aléatoire situé entre les poids des deux parents.
11. On veut faire en sorte que les animaux aient un identifiant unique, donné par un entier. La première instance de classe `Animal` générée recevra le numero 1, la deuxième le numero 2, etc... Proposer un moyen de faire ça.

2 Des Zoos

1. Dans un instant on va vouloir mettre les animaux dans des enclos (un enclos sera juste un tableaux d'animaux). Cependant toutes les espèces ne peuvent pas cohabiter. Par exemple un crocodile ne peut pas être dans le même enclos qu'une gazelle. On veut donc définir un ensemble de couples incompatibles. Proposer une structure de données pour stocker la liste de ces incompatibilités. Où doit-on stocker cette structure, et de quelle façon ? Écrire pour la classe `Animal` une méthode qui prend deux animaux en arguments et teste s'ils sont compatibles.
2. On définit maintenant la classe `Enclos`. Un enclos contient un ensemble d'animaux (représenté par un tableau de taille fixée à l'initialisation). Définir cette classe avec un constructeur permettant d'initialiser la taille de l'enclos, et lui donner une méthode permettant de rajouter un animal. Bien sûr cette méthode sera chargée de vérifier qu'il y a une place libre et que l'animal est compatible avec les animaux déjà présents.
3. On veut définir maintenant la classe `Zoo`. Elle possède comme attributs
   — un attribut `ville` de type chaîne de caractères,
   — un attribut `contenu` de type tableau d'`Enclos`.
   Écrire une méthode permettant de rajouter un animal au zoo si c'est possible (il existe un enclos dans lequel il peut s'insérer).
4. Écrire une méthode qui permet de nourrir tous les animaux du zoo qui ont plus de 5 en faim.
5. Écrire une méthode `passeUnJour` qui fait se dérouler un jour : tous les animaux augmentent en âge, voient leur faim augmenter de 2, ceux qui dépassent 10 meurent, et tous les animaux d'un enclos qui peuvent se reproduire (conditions de la question 10 de l'exercice précédent + une place libre dans l'enclos) le font.
6. On voudrait pouvoir spécifier des caractéristiques propres à chaque espèce (âge de maturité sexuelle, nombre moyen de petits dans une portée, espérance de vie, ...) de façon à ce que la méthode de la question précédente soit adaptée aux espèces. Proposer une solution.

3 Température

Le but de cet exercice est d'écrire une classe représentant la température. Les trois unités possibles seront "Kelvin", "Celsius" ou "Fahrenheit".

1. Définir une classe `Temperature`, décrite par un `double` représentant la température, et un `String` représentant l'unité. Définir un constructeur initialisant un objet `Temperature` à zéro Kelvin.
2. Définir un deuxième constructeur prenant en argument un `double` et un `String` et initialisant la température correspondante.
3. Définir un troisième constructeur prenant en argument une `Temperature` et initialisant une copie de celui-ci.
4. Définir des méthodes permettant d'afficher et de modifier chaque élément d'une `Temperature`.
5. Definir une méthode `conversionKC` convertissant une température donnée en Kelvin en une autre donnée en degrés Celsius, et ne faisant rien si la température initiale n'était pas en Kelvin. On rappelle la formule $T_K = T_C + 273.15$.
6. De même, définir une méthode `conversionCF` convertissant une température donnée en degrés Celsius en une autre donnée en degrés Fahrenheit, et ne faisant rien si la température initiale n'était pas en degrés Celsius. On rappelle la formule $T_F = 9/5 * T_C + 32$.
7. Comment tester l'égalité de deux `Temperature`s (même valeur et même unité) ?
8. Définir une méthode `plusGrande` permettant de comparer deux `Temperature`s.