TP10

Exceptions personnalisée

Animaux et zoo

1. On vous demande de définir une classe `Animal` possédant deux attributs : un nom et un booléen indiquant s'il est blessé ou non.
2. Définissez une classe `Zoo` ayant comme attributs un nom et une liste d'animaux.
3. Définissez une méthode `accueillir` permettant d'accueillir des animaux dans ce zoo.
4. Définissez une méthode `soigner` prenant un Animal en paramètre et permettant de soigner cet animal. Attention, dans les cas où l'animal n'est pas dans le zoo et dans celui où l'animal n'est pas blessé, vous devrez lever une exception personnalisée.
5. Ecrivez une classe `ExecutableZoo` dans laquelle vous devez :

- créer une girafe en pleine forme et un lion blessé ;
- créer un zoo et y accueillir la girafe, le lion et un singe blessé.
- soigner le lion ;
- soigner la girafe ;

Interface et Héritage

Interface Contenant

Dans cet exercice, on va définir une interface `Contenant<T>` représentant la notion de quelque chose qui peut contenir des objets de type `T`.

Ainsi, un `Contenant<T>` est un objet qui a une méthode booléenne `contient` prenant en entrée un objet de type `T`.

1. Écrivez l'interface `Contenant<T>`.
2. Écrivez une classe `Couple` représentant un couple d'entiers et implémentant l'interface `Contenant<Integer>`. Le Couple *contient* l'entier x si l'une ou l'autre des deux valeurs du couple est égale à x.
3. Écrivez une classe `GestionContenants` ayant une méthode statique `contiennentTous` prenant en paramètre un ArrayList<Contenant<T>> conts et un T elem et renvoyant vrai si elem est contenu dans tous les Contenant de conts.
4. Écrivez une classe `Ensemble` représentant un ensemble d'entiers et implémentant l'interface `Contenant<Integer>`. Dans cette question, votre classe contiendra un `Set<Integer>` pour stocker les entiers.

5. Dans un exécutable, créez un `ArrayList<Contenant<Integer>>` dans lequel vous ajouterez le couple (0, 1), l'ensemble {0, 1, 2, 3, 4} et le couple (0, 2). Vérifiez que la méthode `contiennentTous` renvoie vrai pour l'entier 0 et faux pour l'entier 1.