JAVA – Projet partie 1
Licence Informatique

14-avril-2006

Travail à faire en binôme
Ce travail est à rendre avant le mercredi 26 avril.

1. On souhaite écrire une classe `Parking` permettant de réserver des places dans un parking.
Le parking est construit avec un nombre de place de parking.
Vous devez créer les exceptions et faire en sorte d'obliger l'utilisateur à rattraper les exceptions levé par les méthodes.
La classe `Parking` devra posséder les méthodes :

- `int bookPlace()` permet de réserver une place de parking, si aucune place n'est disponible, la méthode devra lever une exception `NoMorePlaceException`
  La méthode devra renvoyée l'index d'une place de parking disponible.
- `void freePlace(int index)` devra libérer une place de parking, si la place de parking n'était pas réservé, la méthode devra lever l'exception Free`PlaceException`
- Une méthode permettant l'affichage de l'occupation du parking, comme ceci :

```
nombre de place total : 3
place 0 : occupé
place 1 : libre
place 2 : occupé
```

- `int freePlaceCount()` renvoyant le nombre de place libre.

2. on veut définir une hiérarchie de classes pour représenter des animaux en vue d'écrire un programme d'aide à la gestion d'un zoo.

On s'intéressera aux propriétés suivantes
- alimentation typique de ces animaux
- poids d'un animal en particulier
- dangerosité habituelle
- le nom de l'animal

La première étape sera définir la hiérarchie de classes pour représenter des :
- Chimpanzé nourriture : 3 kg de banane, pas dangereux, pas protégé, poid max 100 kg
- Orang-outan : 3 kg de banane, pas dangereux, pas protégé, poid max 500 kg
- Lion : 3 kg de viande, dangereux, pas protégé, poid max 200 kg
- Tigre : 4 kg de viande et une jambe de porc, dangereux, pas protégé, poid max 150 kg
- Boa : 1 poulet, dangereux, protégé, poid max 50 kg

1. Créez les classes, classes abstraites ou interfacesAnimal, Mammifere, Singe, Felin, Serpent ainsi que les classes citées ci-dessus.
   Attention vous devez impérativement chercher à limiter la taille des objets de ces classes (nombre de membres). Ne déclarer un membre que si nécessaire

2. Ecrivez les constructeurs des classes nécessaires.

3. Ajouter les fonctions suivantes :

```
String getName()
double getWeigth()
boolean isDangerous()
String getFeedingInfo()
```

4. ajouter la méthode toString() permettant d'obtenir une description complète de l'animal. Cette description contiendra au minimum toutes les infos décrites ci-dessus.

5. Dans le cas du Boa, la méthode doit en plus spécifier que c'est une espèce protégée (an ajoutant un préfixe espèce protégée à la description.

6. vérifier si vous avez correctement utilisé les interface et les classes abstraites. Justifiez leur utilisation éventuelle.
   Avez-vous correctement utilisé private, protected, final ? Justifiez.

7. écrivez une classe de Main de test qui :
- crée 2 Boa, 1 Lion, 1 Oran-outang et les ajoute dans un tableau d'Animal
- Parcours ce tableau pour afficher la description des animaux

8. On veut ajouter une méthode getMaxWeight() qui renvoie le poids maximum d'un type d'animal donné.
   - quels choix avez-vous pour implémenter cette fonction ?
   - comment doit-on faire si l'on veux garantir qu'aucun animal ne puisse avoir un poid supérieur au poid max de son type.
   - Implémenter la fonction avec cette solution,
   (lire le sujet jusqu'au bout SVP).

9. écrivez une classe Zoo qui contiendra un ensemble d'animaux (sans limite de taille). Cette classe doit, au minimum, permettre :
- void add(Animal a) pour ajouter un animal
- String getShortListing() pour obtenir une description courte des animaux du zoo : 1 ligne par animal : son nom et le type d'animal (pensez à la méthode getClass() de Object)
- dans un deuxième temps, ordonner cette liste d'après le nom de l"animal

10. pour aider au nettoyage des cages, on veut savoir combien de temps est nécessaire pour nettoyer la cage d'un animal.
    On a, en général, cette information par la formule : sqrt(poidsAnimal) * 5
    quand un animal est dangereux, le temps est multiplié par 2
    pour les espèces protégées, il faut ajouter 10 minutes par animal.
    Ecrire la méthode double getCleaningTime() qui renvoie cette information.

11. Pour aider à la gestion des approvisionnements, on veut récupérer la "liste des courses" pour nourrir tous les animaux.
    Ecrire la méthode qui renvoie la liste des feedingInfo

12. on suppose que getFeedingInfo pour les lions renvoie "10 kg de canard" et que getFeedingInfo pour les tigres renvoie la même chose.
    Modifier la méthode précédente pour qu'elle regroupe les entrées identiques en préfixant simplement par le nombre d'entrées. Dans notre exemple, si on a un lion et deux tigres, on veut obtenir

    3 x 10 kg de canard
    ...

13. on suppose que les petits animaux mangent moins que les gros ! getFeedingInfo renvoie la nourriture nécessaire pour un animal de poids max. constituer la liste en prenant en compte le poids de chaque animal (exemple : 1 lion de 100kg(poids max=200) => ½ part, 1 tigre de 120 kg (poids max 200) => 0,6 part, 1 tigre de 200 kg => 1 part)

    2,1 x 10 kg de canard

    ...