TD 4: Gestion d'un zoo

notions : Héritage, classes abstraites, Polymorphisme, Liaison dynamique

1 Les animaux

On veut créer une application permettant la gestion d'un zoo. Les fonctionnalités indispensables sont l'affichage de l'animal et le calcul du coût de nourriture par jour.

Ce zoo contient des espèces différentes, avec les propriétés suivantes :
- Chimpanzé : Végétarien, 3kg de fruits/jour, cri : hurlement.
- Autruche : Végétarien, 5kg de fruits/jour, cri : beuglement.
- Aigle : Carnivore, 1 kg de viande, envergure, cri : sifflement.
- Orque : Carnivore, 70 kg de viande, cri : sifflement.
- Tigre du Bengale : Carnivore, 4 kg de viande, nombre de rayures, cri : feulement.
- Tigre Blanc : Carnivore, 4 kg de viande, nombre de rayures, cri : feulement.

Chaque animal a un nom qui lui est propre. Le calcul du coût est obtenu par :
- pour un carnivore : $(quantite * 10)^2 + 100$
- pour un végétarien : $1.2 * log((quantite + 5) * 2 + 1)$

1.1 Diagramme de classes

Représenter graphiquement par un diagramme de classes (le diagramme d'héritage) les différentes classes utiles pour représenter ces animaux. Précisez les champs et les prototypes de méthodes, en respectant l'encapsulation (accesseurs, mutateurs, protections des champs et des méthodes). Votre diagramme de classes est un arbre comportant au moins 3 niveaux, avec des classes abstraites et des méthodes abstraites. Quelques éléments à prendre en compte pour construire votre représentation.
- quels sont les variables ou attributs qui sont communes à tous les types d'animaux ?
- quels sont les variables ou attributs qui sont spécifiques à certains types d'animaux ?
- quels sont les éléments qui sont des constantes et non des variables pour un type d'animal ?
- quels sont les comportements ou méthodes que doivent posséder tous les animaux ?
- parmi ceux-ci, quels sont ceux qui sont parfaitement définis et ceux qui ne le sont pas au niveau des animaux ?

1.2 Classes en Java

Écrire en Java ces classes, en respectant les notions d'encapsulation (privé ou public) et la notion de réutilisabilité. Ces classes doivent être définies dans le package `zoo.animal` (File → new → package). Elles comporteront au moins :
- Un constructeur adéquat permettant de construire un objet avec les paramètres nécessaires. Utilisez correctement les constructeurs des classes mères.
- Une méthode permettant l'affichage du contenu de l'objet. La méthode `String toString()` sera utile à cet effet.
- Le calcul du coût de nourriture.
- La méthode `boolean equals(Object)` qui indique si 2 animaux sont identiques ou nom.

1.3 Un programme de test

Ecrire une classe TestAnimaux contenant une méthode main construisant un objet aigle *Grandaigle* et un chimpanzé *Chita*, les affichant et calculant le cout de chacun d'eux. Créer un autre aigle de nom *Grandaigle*

et comparer les 2 aigles d'une part, l'aigle et le chimpanzé d'autre part en utilisant la méthode `equals` et l'opérateur `==`.

2 Le Zoo

Le zoo est défini par son nom et par l'ensemble des animaux qu'il possède. Il faut pouvoir créer le zoo, ajouter un nouvel arrivant (animal) au zoo, supprimer un animal du zoo, afficher l'ensemble des animaux actuellement présents dans le zoo, calculer le coût de la nourriture du zoo. L'ensemble des animaux sera stocké dans un tableau d'animaux.

2.1 La classe Zoo

Écrire en Java une classe Zoo dans le package `zoo` comportant :
- Un constructeur adéquat, avec le nom du zoo et le nombre de places disponibles.
- une méthode d'ajout d'un animal. Le prototype est : `void ajoute(Animal ref)`. L'animal ref est ajouté dans la première place disponible du tableau d'animaux. Cette place est caractérisée par une référence `null`.
- une méthode pour supprimer un animal à partir de son nom. Le prototype est : `void supprime(String nom)`. Ne pas oublier de rendre la case du tableau disponible ensuite en la mettant à la référence `null`.
- une méthode pour supprimer un animal à partir de sa référence. Le prototype est : `void supprime(Animal ref)`
- le calcul du nombre d'animaux présents dans le zoo : c'est le nombre de cases du tableau des animaux qui n'est pas `null`.
- le calcul du coût total de la nourriture nécessaire par jour pour les animaux.
- Une méthode permettant l'affichage du zoo : son nom, le nombre d'animaux et tous les animaux le composant. La méthode `String toString()` sera utile à cet effet.

2.2 Un programme de test

Écrire la fonction main qui effectue les actions suivantes :
- Création du zoo « Minatec ».
- Ajout du tigre du Bengale « Fantôme » , de poids 120kg, 40 rayures.
- Ajout de l'autruche « Ann », de poids 50kg.
- Ajout du chimpanzé « Chita » de poids 30kg.
- Ajout de l'aigle « Roquette », d'envergure 200cm, de poids 5kg.
- Ajout de l'orque « Azog », de poids 9000kg.
- Affichage du coût de nourriture du zoo.
- Suppression du tigre « Fantôme ».

3 L'approche par délégation

On peut aussi gérer le régime alimentaire des animaux par délégation en créant des classes spécifiques, en particulier dans le cas où le régime regrouperait un grand nombre d'actions et de données à gérer. Un régime alimentaire est défini par le type de nourriture et la méthode de cout de calcul. L'animal délègue ainsi le calcul à une autre classe que la sienne. De la même manière qu'auparavant, la classe Regime sera abstraite. Les classes de niveau inférieur définiront la bonne méthode de calcul de cout ainsi que la fonction d'affichage du régime associé.

3.1 Classes Regime et dérivées

Effectuer les modifications suivantes :
- Création de la classe Regime dans le package `nourriture` et des sous classes Legume et Viande.
- Modification des classes représentant les animaux : il existe maintenant un attribut `regime` propre à chaque animal. Modifier en particulier les constructeurs des animaux.