Modélisation d'un zoo

L2 POO - TP 4-5

1 Introduction

Le zoo de La Flèche possède plus de 1 600 animaux correspondant à plus de 1000 espèces animales. Il devient indispensable de s'équiper d'une gestion informatique pour aider au mieux le personnel du zoo. Vous êtes donc missionnés par le zoo pour modéliser les animaux présents. Cependant, le personnel du zoo ne souhaite pas passer à un système complet et préfèrent d'abord tester cette nouvelle gestion sur seulement neuf espèces.

Dans un premier temps, nous limiterons la composition du **Zoo** aux **Félins** (**Lion**, **Tigre** et **Panthere**), aux **Oiseaux** (**Aigle**, **Faucon** et **Perroquet**) et aux **Cervidés** (**Cerf**, **Élan** et **Daim**). Il est important de préciser que tous ces **Animaux** ont un `nom` et un `age` arrondi à l'entier.

2 Création de la hiérarchie de classes et des fonctions de base

Question 1 : Commencez par modéliser **à l'écrit** une hiérarchie de classes permettant de répondre à la demande du personnel du zoo. Avant de passer à la question suivante, faites valider votre hiérarchie de classes par l'enseignant.

Question 2 : Implémentez dans Eclipse la hiérarchie de classes. (*Rappel :* Eclipse propose de générer directement les *Getters* et *Setters* d'une classe en passant par `Source` > `Generate Getters and Setters ...` après avoir défini les attributs).

Question 3 : Nous souhaitons commencer par créer une fonction qui permet à tout moment d'afficher un animal. Nous avons besoin de redéfinir la fonction `toString()` dans la classe **Animal** pour obtenir un affichage similaire à l'exemple ci-dessous :

```
// affichage pour un lion de 2 ans dont le nom est Simba
"Simba a 2 ans."
```

Question 4 : Créez une classe **Zoo** qui possède un `nom` et un `tableau de type Animal`. Il est à noter qu'un zoo a évidemment une capacité d'accueil maximum ! Cependant, il est rare que le zoo soit plein, il faut donc être capable de dire combien d'animaux sont présents.

Vous aurez donc besoin d'une fonction `ajouterAnimal()` et `supprimerAnimal()` qui prennent une instance de la classe **Animal** en paramètre.

Créez ensuite une instance de la classe **Zoo** pour modéliser le zoo de La Flèche. Enfin, remplissez votre instance de la classe **Zoo** avec des animaux des neufs espèces (au moins un animal par espèce). Vous trouverez dans la Table 1 quelques idées mais vous êtes libres de créer les animaux que vous voulez.

De même, dotez votre classe **Zoo** d'une méthode d'affichage qui permettra d'afficher le zoo et ses animaux. Affichez ensuite votre zoo.

Nom	Simba	Mufasa	Shere Khan	Bagheera	Bambi	Rutt	Féline	Sitka
Espèce	Lion	Lion	Tigre	Panthère	Cerf	Élan	Daim	Aigle
Age	2	18	14	7	1	10	4	14
Nom	Iago	José Carioca		Hayabusa			Nawras	Nathalie
Espèce	Perroquet	Perroquet		Faucon			Lion	Tigre
Age	4	7		7			4	3
Nom	Alexandre	Marie	Simon	Cédric	Hakim	Alain	Benoit	Rebecca
Espèce	Lion	Tigre	Panthère	Cerf	Élan	Daim	Aigle	Faucon
Age	2	5	6	7	1	11	15	2

Table 1 – Quelques exemples d'animaux pour peupler le zoo

Question 5 : Le personnel du zoo vous remercie pour ce début de programme mais souhaiterait que l'affichage indique aussi l'espèce de l'animal qui s'affiche. Modifiez votre code pour prendre en compte cette modification. Pour cela, nous souhaitons réutiliser le code déjà écrit et ajouter un autre String à la réponse de la fonction `toString` de la classe **Animal**. Effectuez cette modification **uniquement** dans les classes **Félin**, **Lion**, **Tigre**, **Panthere**, **Cervidé**, **Oiseau**, **Aigle** et **Faucon**. Vérifiez que vous obtenez un affichage semblable à cela :

```
// affichage pour un Lion de 2 ans dont le nom est Simba
"Simba a 2 ans. C'est un Felin. C'est un Lion."

// affichage pour un Cerf de 1 an dont le nom est Bambi
"Bambi a 1 an. C'est un Cervide."

// affichage pour un Aigle de 14 ans dont le nom est Sitka
"Sitka a 14 ans. C'est un Oiseau. C'est un Aigle."
```

Afficher de nouveau votre zoo, que constatez-vous ?

2.1 Analysons un peu le zoo

Question 6 : Nous sommes maintenant capable d'afficher correctement les **Animaux** présents dans le zoo. Pour compléter, nous souhaitons afficher quelques informations telles que l'**Animal** le plus âgé, l'âge moyen des **Animaux** et les enfants (i.e les **Animaux** de 2 ans ou moins). Écrivez dans la classe **Zoo** les fonctions `doyen()`, `ageMoyen()` et `enfants()`.

Question 7 : Dire qu'un enfant est un animal de 2 ans maximum n'est pas totalement vrai car tous les **Animaux** ne vivent pas le même temps (par exemple les tortues vivent plus de 100 ans alors que les orques vivent en moyenne 29 ans). Le personnel du zoo vous demande donc de corriger la fonction `enfants()` en se basant cette fois sur un seuil établi à 25% de la longévité de l'espèce. Pour cela, il est nécessaire de doter votre programme d'une nouvelle fonction `getLongevite()` qui retournera par espèce une constante correspondante à la longévité moyenne de cette espèce. Il est important que cette fonction soit déclarée au niveau de la classe **Animal** sinon nous ne pourrons pas l'utiliser puisque notre zoo contient une liste d'**Animaux**. Cependant, il est difficile de se prononcer sur la longévité d'un animal sans savoir son espèce. Cela veut donc dire qu'une partie de notre hiérarchie de classes doit être `abstraite`. Ajoutez les mots-clés `abstract` aux endroits nécessaires si cela n'est pas déjà fait.

Les soigneurs vous donnent les informations suivantes : les lions vivent en moyenne 12 ans ; les tigres 23 ans et les panthères 15 ans. Quant aux cervidés, leur longévité est environ de 20 ans. Les perroquets vivent 45 ans ; les aigles 14 ans et les faucons 13 ans.

2.2 Organisation de spectacles d'oiseaux

Question 8 : Le personnel du zoo commence à s'habituer à votre logiciel et souhaite alors l'améliorer. Le zoo a pour habitude d'organiser des spectacles où intervient uniquement un oiseau (ce ne sera pas toujours le même). Le personnel du zoo vous demande donc d'ajouter la gestion de la sélection de l'**Oiseau** qui participera au prochain spectacle. Cet **Oiseau** sera celui ayant participé au moins de spectacles, soit l'oiseau le moins fatigué. Cela semble relativement simple puisque cela revient à compter chaque participation d'un oiseau à un spectacle. Pour cela, vous aurez besoin d'une fonction `creerSpectacle()` qui prend en entrée l'**Oiseau** qui participe au spectacle et affiche un message tel que celui-là :

```
// affichage d'un spectacle avec la participation d'un Aigle dont le nom est Sitka
"Sitka participa a un spectacle."
```

De plus, vous aurez besoin d'une fonction `trouverOiseauLeMoinsFatigue()` qui retourne l'**Oiseau** du zoo ayant participé au moins de spectacles. Cela pose cependant un problème puisque votre **Zoo** utilise un tableau d'**Animaux**. Il faut donc être capable de déterminer pour chaque **Animal** s'il est un **Oiseau** ou pas. Pour cela nous ferons appel au mot-clé `instanceof` de Java (si vous ne connaissez pas ce mot-clé, la Javadoc est là pour ça !). Si plusieurs oiseaux possèdent le même nombre de participations à un spectacle, vous êtes libre de choisir celui que vous voulez garder.

Question 9 : Mettez à jour la méthode `toString` pour afficher le nombre de participations à un spectacle pour un **Oiseau**.

```
// affichage pour un Aigle de 14 ans dont le nom est Sitka
```

```
"Sitka a 14 ans. C'est un oiseau qui a participe a 3 spectacles. C'est un aigle."
```

3 Pistes d'améliorations (facultatif)

Question 10 : Placez ces lignes de code dans votre main(). Quelles lignes ne fonctionnent pas ? Pourquoi ? Est-ce normal ?

```
Animal a1 = new Animal("Cedric", 12) ;
System.out.println(a1);

Animal a2 = new Lion("Simon", -1) ;
System.out.println(a2);

Felin f1 = new Lion("", 12) ;
System.out.println(f1);

Felin f2 = new Lion("Cedric", 100) ;
System.out.println(f2);

Lion l1 = new Lion("Alexandre", 5) ;
System.out.println(l1);
```

Question 11 : Implémentez une fonction pour déterminer quelle espèce est la plus représentée ? De même, quelle est la famille d'espèces (**Félin**, **Cervidé** ou **Oiseau**) la plus représentée ?

Question 12 : Ajouter à la classe **Animal** un attribut de classe de type **Animal** et nommé `parent`. Cet attribut représentera un des deux parents. Pensez à mettre à jour vos constructeurs avec paramètres pour permettre de construire un **Animal** en indiquant son parent en plus des autres paramètres. On notera que cet attribut peut être nul si le parent de l'animal n'est pas présent dans le zoo ! Vous pouvez alors ajouter une méthode `getParent()` dans la classe **Animal** pour avoir accès au parent d'un **Animal**.

Exemple :

```
// Cedric n'a pas de parent dans le zoo
Animal a1 = new Lion("Cedric", 12) ;

// Cedric est parent de Marie
Animal a2 = new Lion("Marie", 3, a1)
System.out.println(a2)
"Marie a 3 ans et son parent est Cedric. C'est un felin. C'est un lion."
System.out.println(a2.getParent())
"Cedric a 12 ans. C'est un felin. C'est un lion."
```

Question 13 : Afin de mieux représenter le zoo, nous allons maintenant modifier la structure même du zoo en ajoutant la notion d'**Enclos**. Un **Zoo** contiendra un certain nombre d'**Enclos** (déterminé lors de la construction du **Zoo**) et chaque enclos contiendra cinq **Animaux** maximum. Implémenter la classe **Enclos**, puis écrivez une nouvelle classe **Zoo** qui ne contiendra plus une liste d'**Animaux** mais d'**Enclos**.

Question 14 : L'implémentation en l'état du **Zoo** permet d'avoir des **Lions** et des **Cerfs** ensemble dans un même **Enclos**. Cela n'est pas réaliste. Modifiez votre code pour n'autoriser l'ajout d'un **Animal** à un enclos si et seulement si il est de la même espèce que les **Animaux** déjà présents dans cet enclos.

Question 15 : Suite à la mise en place des enclos, les fonctions telles que `doyen()` ou `enfants()` ne sont plus utilisables car elles utilisent un tableau d'instances de **Animal**. Une solution simple est d'implémenter une fonction `getAnimaux()` qui étant donné une tableau d'enclos, retourne la liste des **Animaux** présents dans l'ensemble des **Enclos**. Implémentez cette fonction.