Cartouche du document

Année : ING 1
Matière : UML
Activité : Travail dirigé

Objectifs

Il s'agit dans ce travail dirigé d'introduire la conception orientée objet à travers

- le concept d'interface pour privilégier la programmation générique à le programmation spécifique
- la réutilisation : Héritage ou Composition
- une première approche des design patterns : méthode de fabrique

Sommaire des exercices

Corps des exercices

1 - Réutilisation : Confusion entre héritage sémantique et héritage fonctionnel

Énoncé :

On essaie dans cet exercice de montrer que l'héritage n'est pas toujours une bonne solution pour la réutilisation. Dans le package java.util la classe Stack a été implémentée en dérivant la classe Vector.

Question 1)

Énoncé de la question

Expliquer en quoi le code suivant :

```
Stack pile = new Stack();
pile.push("Bas de la pile");
pile.push("Haut de la pile");
pile.insertElementAt("Perdu", 0);
while (!pile.empty()) {
System.out.println(pile.pop());
}
```

viole le fonctionnement d'une pile.

Solution de la question

Les deux premiers éléments ont été placés dans la pile avec le principe LIFO mais pas le troisième. Quand on exécute le code on a l'affichage suivant :

```
Haut de la pile Bas de la pile Perdu
```

alors qu'on aurait du avoir

```
Perdu Haut de la pile Bas de la pile
```

Question 2)

Énoncé de la question

En quoi a-t-on fait un héritage fonctionnel et non pas un héritage sémantique.

Solution de la question

On a utilisé la réutilisation de la classe Vector en tenant compte de certaines des ce fonctionnalités comme l'ajout d'un élement, la suppression d'un élement et la récupération d'un élément. Le principe LIFO a été pris en compte dans l'API Stack mais n'est pas respecté dans l'API Vector

Question 3)

Énoncé de la question

Pourquoi ne peut-on pas dire qu'une pile **est un** vecteur ?

Solution de la question

Une pile n'est pas un vecteur car

- on ne peut pas ajouter un élément à n'importe quel endroit : la méthode **add(int index, E element);** ;
- on ne peut pas supprimer des élements dans un ordre quelconque : la méthode **remove(int index);**

Question 4)

Énoncé de la question

Pproposer une implémentation de la classe Pile en utilisant la classe Vector par composition ?

Solution de la question

```
import java.util.*;
class GoodStack {
 private Vector v;
 public GoodStack() {
  v = new Vector();
 }
 public boolean empty() {
  return v.capacity() == 0;
 }
 public Object peek() {
  return v.elementAt(0);
 }
 public void push(Object o) {
  v.add(o);
 }
 public Object pop() {
  if( ! empty() ) {
   Object o = v.elementAt(0);
   v.remove(0);
   return o;
  }
  else
```

```
  return null;
 }
 public int search(Object o) {
  return v.indexOf(o);
 }
}
```

2 - Réutilisation : La bonne utilisation de l'héritage

Énoncé :

On montre dans cet exercice que l'héritage est bon choix pour la réutilisation quand la relation entre de la forme ObjetFille **est un (ou est une sorte de)** ObjetMere. On suppose que la classe Vehicule a été implémentée. Definition : un véhicule est un moyen qui permet à un ou plusieurs être humains de se déplacer d'un endroit A à un endroit B.

Question 1)

Énoncé de la question

Quel sont les liens sémantiques entre un véhicule et

- une voiture, une moto, un vélo, un cheval, un élephant, un rain, un avion, un bateau et un radeau
- un chauffeur

Solution de la question

Une voiture, une moto, un vélo, un cheval, un élephant, un train, un avion, un bateau et un radeau **sont des sortes** de véhicules.

Un chauffeur **utilise** un véhicule.

Question 2)

Énoncé de la question

On suppose que dans la classe Vehicule on a implémenté l'opération **void deplaceToi()**. On constate un peu plus tard que le déplacement dépend du vent et que la signature devient **void deplaceToi(int vent)**.

Que faut-il recoder si

- 1) si on a réutilisé par héritage la classe Vehicule pour les classes Voiture, Moto, Velo, Cheval, Elephant, Train, Avion, Bateau et Radeau.
- 2) si on a réutilisé par composition la classe Vehicule pour les classes Voiture, Moto, Velo, Cheval, Elephant, Train, Avion, Bateau et Radeau.

Solution de la question

Par héritage, il suffira de recoder l'opération deplaceToi une seule fois dans la classe Vehicule. Par compostion, il faudra recoder l'opération deplaceToi dans chacune des classes Voiture, Moto, Velo, Cheval, Elephant, Train, Avion, Bateau et Radeau.

Question 3)

Énoncé de la question

Qu'est ce qui nous assure dans ce cas que l'héritage est une bonne solution ?

Solution de la question

Le lien qui relie la classe Vehicule d'une part et les classes Voiture, Moto, Velo, Cheval, Elephant, Train, Avion, Bateau et Radeau d'autre part est un lien sémantique. Tout changement de comportement de la classe Vehicule doit être automatiquement reporté sur les classes Voiture, Moto, Velo, Cheval, Elephant, Train, Avion, Bateau et Radeau.

Question 4)

Énoncé de la question

Reprenons le lien sémantique **un chauffeur utilise un véhicule**. Cela a pour conséquence qu'un véhicule peut ne pas avoir de chauffeur. On peut effectivement envisager qu'un véhicule se déplace automatiquement sans chauffeur.

Expliquer pourquoi la composition est un meilleur choix.

Solution de la question

Par composition un véhicule contient une référence sur un chauffeur. Cette référencec pourra donc être nulle. Si la classe Vehicule hérite de la classe Chauffeur alors un objet Vehicule contiendra un objet Chauffeur même dans le cas où il n'y a pas de chauffeur.

3 - Interface : Passer de la programmation spécifique à la programmation générique

Énoncé :

Dans le projet Mini Système Expert, on a le diagramme de classe suivant :

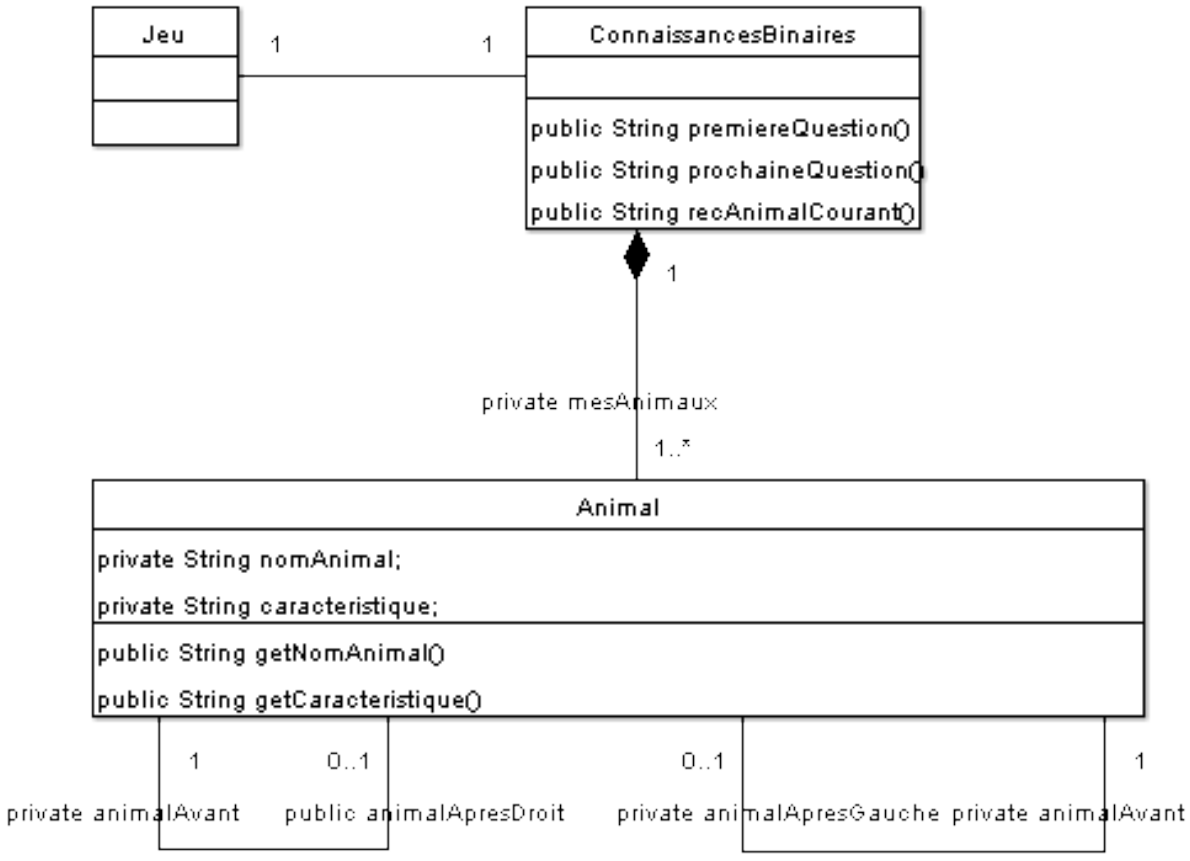

Diagramme de classes spécifique : Jeu, Connaissances et animaux

Question 1)

Énoncé de la question

Ce diagramme de classes est spécifique à certain mode de structuration des connaissances. On vous demande de définir un autre diagramme qui permet de découpler la classe Jeu et la classe ConnaissancesBinaires afin d'envisager un autre mode de structuration des connaissances.

Solution de la question

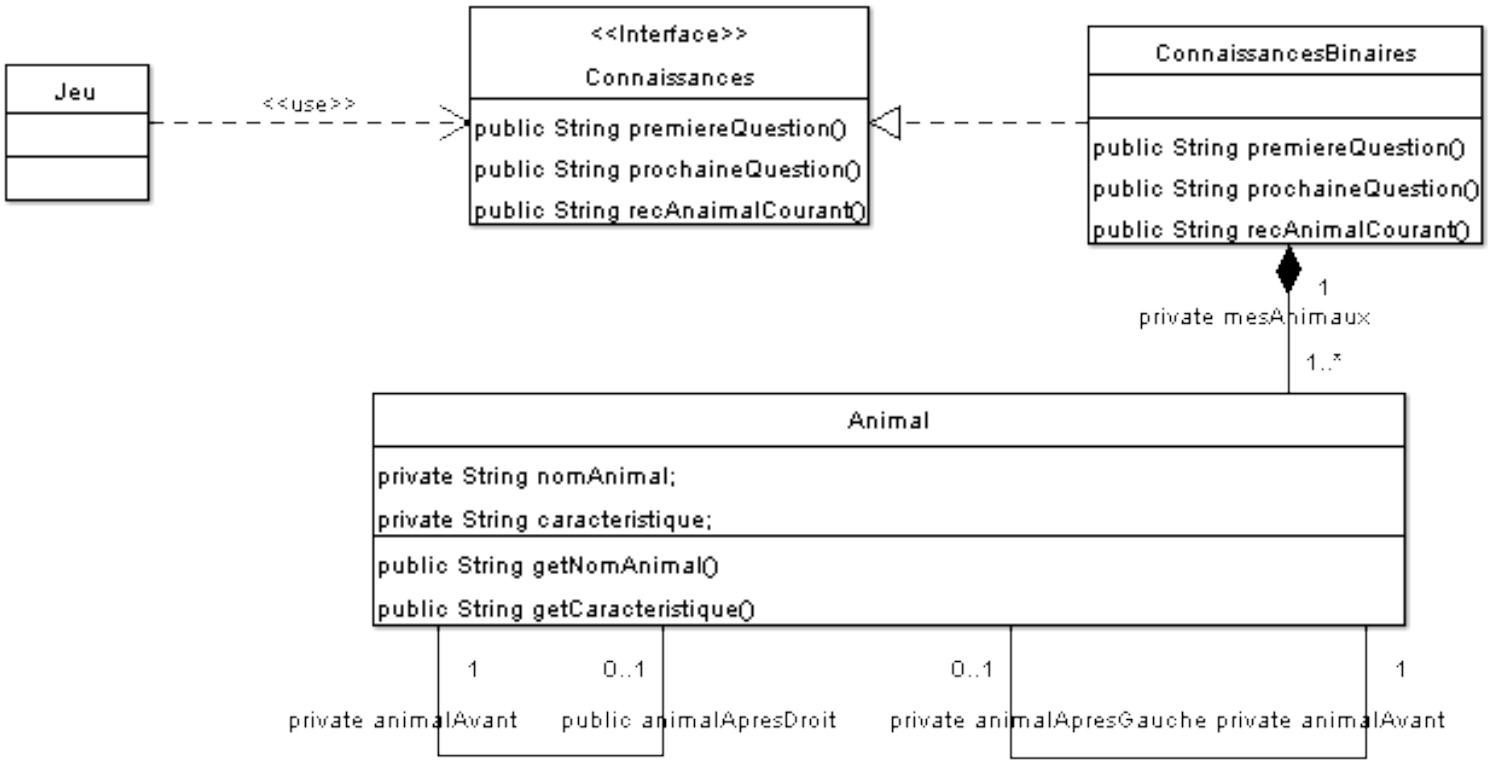

Diagramme de classes générique : Jeu, Connaissances et animaux

Question 2)

Énoncé de la question

Que faut-il ajouter dans le diagramme de classes pour que l'on puisse créer l'objet de connaissances sans que le client Jeu connaisse le type de cet objet ?

Solution de la question

On applique le pattern **Méthode de fabrique** en créant une classe qui hérite de la classe Jeu et qui implémente l'opération de fabrication de l'objet de connaissances.