

Généricité paramétrique

Travaux dirigés et pratiques - Typage statique (UE UMINIM202)

Ce document constitue un support de réflexion sur la généricité paramétrique en C++ et Java. Il est conseillé d'écrire des diagrammes UML pour mieux visualiser les types construits.

1 Espèces et animaux

On décrit séparément des espèces d'animaux et leurs membres (par deux classes). Les espèces sont décrites par leur nom latin. Les membres (individus) d'une espèce sont décrits par leur nom et leur âge. Imaginez une modélisation simple et une modélisation dans laquelle les espèces sont passées comme paramètres aux individus. Créez quelques objets. Dans les deux modélisations, peut-on dire facilement qu'un individu canari est un individu oiseau ?

2 Vétérinaire

Nous nous intéressons aux classes de vétérinaires spécialisés pour certaines espèces d'animaux. Un vétérinaire a un nom et peut admettre comme patients des animaux de l'espèce qu'il est habilité à soigner (prévoir une méthode d'admission d'un patient). Créez un vétérinaire spécialisé pour les oiseaux, un oiseau, admettre cet oiseau comme patient pour ce vétérinaire. Etudiez deux solutions : passer la classe espèce ou la classe membre de l'espèce en paramètre de Vétérinaire.

- un vétérinaire spécialisé pour les oiseaux est-il un vétérinaire pour les gallinacées ?
- un vétérinaire spécialisé pour les gallinacées est-il un vétérinaire pour les oiseaux ?
- un vétérinaire spécialisé pour les oiseaux peut-il admettre une poule comme patiente ?
- un vétérinaire spécialisé pour les gallinacées peut-il admettre une colombe comme patiente ?

On cherche à savoir si un vétérinaire porte le même nom qu'un autre vétérinaire (éventuellement spécialisé pour une autre espèce). Donnez une méthode d'instance et une méthode de classe pour réaliser ce traitement. On veut limiter le nombre de patients possibles. Imaginez une solution avec et sans utilisation de la généricité paramétrique.

On veut comptabiliser le nombre de vétérinaires spécialisés pour une espèce et le nombre de vétérinaires en général. Trouvez des solutions avec et sans variables de classe.

On veut passer comme paramètre une fonction de comparaison des animaux pour déterminer lequel est le plus prioritaire pour les soins (différentes fonctions sont imaginables, utilisant l'âge, l'ancienneté du patient chez le vétérinaire, la gravité du problème, etc.). Comment s'y prendre ?

Peut-on donner une forme particulière à la méthode soigne des vétérinaires spécialisés dans les éléphants ?

Définissons une sous-classe représentant les vétérinaires agrémentés par un organisme d'élevage selon les méthodes biologique. Un vétérinaire agrémenté spécialisé pour les gallinacées est-il un vétérinaire spécialisé pour les gallinacées ? Inversement ?

Comment définir le type des vétérinaires traitant des animaux d'une certaine espèce et habitant une certaine zone ?

3 Etablissement animalier

Un établissement animalier spécialisé est destiné à recevoir des animaux d'une espèce. Il est constitué de plusieurs unités, chacune destinée à recevoir une sous-espèce de l'espèce principale. Ecrivez une méthode d'ajout d'une unité et une méthode d'ajout d'un habitant dans une unité. On peut déplacer les animaux d'une unité B à une autre unité A si B abrite des animaux d'une sous-espèce de ceux abrités par A. Ecrivez une méthode d'instance `deplaceAnimauxDepuis(unitéOrigine)` qui permet de déplacer depuis l'unité passée en paramètre vers le receveur du message, puis la méthode inverse `deplaceAnimauxVers(unitéDestination)` qui permet de déplacer les animaux depuis le receveur du message vers l'unité passée en paramètre. On doit pouvoir mettre les animaux d'une unité de gallinacées dans une unité d'oiseaux (mais pas l'inverse).

4 Fiche de liaison

On définit à présent une fiche de liaison entre un vétérinaire et un établissement animalier ou entre un vétérinaire et une unité. Faites dériver la classe fiche de liaison d'une classe paire d'éléments. Imaginez les contraintes qui sont nécessaires pour le bon fonctionnement des choses.

5 Polymorphisme F-borné

Imaginez une contrainte faisant réapparaître le paramètre de type.