TP2 - Mise à niveau Java

E. Bruno – `bruno@univ-tln.fr`

Objectif

L'objectif de ce TP est d'illustrer les concepts de base du langage Java que vous devez maîtriser. Il permettra une première mise en pratique de l'utilisation de SVN et de trac. Pour cela, vous lirez le sujet en entier et mettrez en place un sitetrac pour suivre l'avancement de vos révisions : mise en place d'une roadmap et de ticket pour chaque question.

Pour tous les exercices, vous devez utiliser l'environnement Eclipse (`http://www.eclipse.org/`). Vous devez écrire la javadoc minimal (`http://java.sun.com/j2se/javadoc/writingdoccomments/`) au fur et à mesure de l'avancement.

La compilation et la génération de la documentation et du fichier jar seront réalisés avec ant (`http://ant.apache.org`).

Il est conseillé d'avoir le documentation de java sous les yeux :
`http://java.sun.com/javase/6/docs/api/`

1 Classes, Instances et Héritage

1.1 Hiérarchie

Dans un paquetage appelé « vivants ». Ecrire les classes Java permettant de décrire une hiérarchie d'animaux : Tous les animaux doivent avoir un nom, ils sont soit des mammifères, soit des oiseaux. Pour le moment les animaux sont des vaches et des aigles.

Créer une classe exécutable Test, dont la méthode main instancie des vaches et des aigles. Avec quel type de références peut-on les manipuler ?

1.2 Constructeurs

En utilisant les constructeurs modifier les classes précédentes pour que la méthode toString() des classes vache et aigle, affiche : Je m'appelle Geronimo et je suis un animal, je suis un oiseau, je suis un aigle Je m'appelle Marguerite et je suis un animal, je suis un mammifère, je suis une vache

1.3 Collections

- `http://java.sun.com/j2se/1.4.2/docs/api/java/util/Collections.html`
- `http://java.sun.com/docs/books/tutorial/collections/index.html`

Nous allons maintenant regrouper les animaux dans des zoos qui sont des collections d'animaux. Créer une classe Zoo (dans un paquetage « structures »), qui possède une méthode ajouteAnimal(), et une méthode afficheToi(). Dans la classe Test, créez deux zoos et peuplez les avec des animaux pour obtenir le résultat suivant :

```
Affichage du zoo : mon Zoo
Je m'appelle Germaine et je suis un animal, je suis un mammifère, je suis une vache
Je m'appelle Geronimo et je suis un animal, je suis un oiseau, je suis un aigle
Je m'appelle Marguerite et je suis un animal, je suis un mammifère, je suis une vache

Affichage du zoo : ton Zoo
Je m'appelle Gertrude et je suis un animal, je suis un mammifère, je suis une vache
Je m'appelle Philomène et je suis un animal, je suis un mammifère, je suis une vache
Je m'appelle Tiki et je suis un animal, je suis un oiseau, je suis un aigle
```

Comment s'assurer que deux animaux qui ont le même nom ne peuvent être ajouté au même zoo ?

1.4 Administration de classes

Comment faire pour qu'une référence vers tous les animaux créés (y compris ceux qui n'ont pas été ajouté à un zoo) soient conservée ? L'ensemble des animaux sera appelé la faune. Où faut-t-il ajouter une méthode afficheToi() qui lister la faune ?

1.5 Héritage Multiple / Interfaces

On souhaite pouvoir distinguer, parmi les animaux, les carnivores et les herbivores. Quel est le problème avec la hiérarchie de classe actuelle ? Une solution à ce problème consiste à utiliser deux interfaces (un carnivore peut manger un autre animal, un herbivore broute). Décrivez ces deux interfaces, et manipuler les aigles et les vaches à travers celles-ci.

1.6 Entrées/Sorties et sauvegarde (serialisation)

– `http://java.sun.com/docs/books/tutorial/essential/io/index.html`

Ajouter à la classe Zoo, une méthode sauve() (doit-elle être de classe ou d'instance ?) qui enregistre un zoo complet dans un fichier dont le nom est passé en paramètre. Ajouter une classe restaure() (doit-elle être de classe ou d'instance ?), qui retourne une référence vers un objet Zoo construit à partir des valeurs trouvées dans un fichier dont le nom est passé en paramètre.

1.7 Livraison du résultat

– `http://java.sun.com/docs/books/tutorial/jar/basics/index.html`

Fabriquer un fichier nature.jar qui contient toutes les classes nécessaires. Exécuter l'application à partir de ce fichier.

1.8 JDBC

Créer une base de données relationnelle contenant des animaux. Ecrire le programme qui crée la faune correspondant aux animaux décrits dans la base de données.

1.9 IHM (facultatif)

Créer une petit IHM permettant d'ajouter des vaches ou des aigles à la faune en précisant leur nom. A chaque ajout, l'affichage de la faune devra être mis à jour.

1.10 Gestion des log (Après une présentation rapide de log4j)

– `http://logging.apache.org/log4j/`

Modifier vos classes pour que les logs sont gérés proprement à l'aide de la classe log4j.

1.11 Test de vos classes avec JUnit (Après une présentation rapide de Junit)

– `http://junit.sourceforge.net/`

Ecrire les classes de test permettant de valider votre implantation avec JUnit.