Fiche d'exercices
Polymorphisme & Collections d'objets
Séances 2 et 3

Ce TD est la suite du TD Animaux. Vous pouvez donc continuer dans le même dossier de travail.

Objectifs :

— Comprendre la notion de polymorphisme de classes
— Créer, gérer une collection d'objets

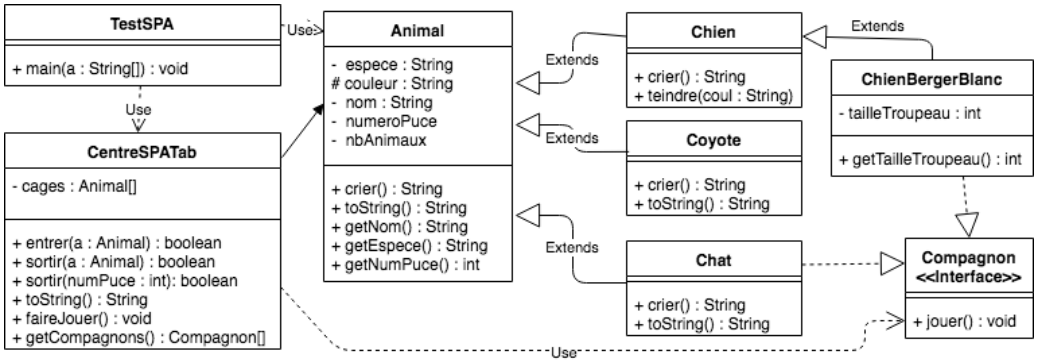

Figure 1 – Diagramme de classes du programme complet, une fois fini

1 Polymorphisme : premières manipulations (30')

Copiez le code suivant dans votre classe de test et répondez aux questions.

```
Animal a = new Chien("Gaston", "tachete");
System.out.println(a.toString()) ;

a = new Chat("Moustache", "blanc") ;
System.out.println(a.toString()) ;

Animal b = new ChienBergerBlanc("Flocon", 100) ;
System.out.println(b.toString()) ;

Chien ch2 = b ;
System.out.println(b.toString()) ;
System.out.println(ch2.toString()) ;

ChienBergerBlanc cb2= ch2 ;
System.out.println(ch2.toString()) ;
System.out.println(cb2.toString()) ;
```

(Q1.1) Pour chacune des 4 variables, quels sont ses types statique et dynamique ? (Notez vos réponses en commentaire dans votre programme.)

(Q1.2) Corrigez ce code pour le rendre fonctionnel.

Remarque : *l'ordre des paramètres dans vos constructeurs est peut-être différent.*

2 Centre SPA (Mise en œuvre du polymorphisme – 90mn)

Un centre SPA est capable d'accueillir des animaux et de les faire adopter. Par ailleurs, les gestionnaires de centre veulent pouvoir connaître à tout moment la liste des animaux hébergés.

(Q2.1) Réalisez une classe `CentreSPATab` avec les fonctionnalités suivantes.

Nous considérerons qu'un centre SPA a une capacité maximale de N animaux qui sera fournie en paramètre au constructeur. Cette classe devrait implémenter les trois méthodes suivantes :

— *entrer(...)* : ajoute un `Animal` au centre. Il sera placé dans la première cage (case) disponible.
— `toString()` : renvoie la liste des animaux présents, avec leur numéro de cage (case).
Rappel : `\n` permet de faire un retour à la ligne dans une chaine de caractères.
— `sortir(a : Animal)` : retire un animal a du centre et renvoie vrai ssi le retrait a eu lieu.

(Q2.2) Si vous n'avez pas testé votre programme : testez le, puis implorez pardon.

(Q2.3) Ajoutez la notion de puce d'identification aux animaux comme suit.

Nous voulons que `tout` Animal ait une puce, donc un numéro de série. Ajouter la propriété `numeroPuce` dans la classe adaptée. Ce numéro de série devra-t-être généré automatiquement à la naissance (création) de tout animal. Faites en sorte que chaque animal ait un numéro **unique** et ajoutez un getter `getNumPuce` permettant de connaitre ce numéro. Pensez à l'intégrer dans la méthode `toString`.

Aide 1 : il existe plusieurs solutions et aucune n'a été vue en cours.

Aide 2 : Informez vous sur static.

(Q2.4) Surchargez la méthode `sortir`, pour qu'il soit aussi possible de récupérer un animal dont on fournit le numéro de puce.

(Q2.5) Modifiez votre méthode `equals` pour qu'elle soit conforme aux règles et intégrez y l'usage de ce numéro.

2.1 Bonus (S'il reste du temps en séance 2)

(Q2.6) Faites une classe `CentreSPAPile` similaire à celle-ci, mais où le retrait d'un Animal entraine le décalage des suivants, pour que les i animaux présents soient toujours dans les cases d'indice 0 à $i-1$.

Ici, la notion de case ne sera plus une analogie aux cages.

(Q2.7) Méthode entrer() : refusez l'ajout si l'animal est déjà présent dans le tableau

3 Les interfaces (60 mn)

Certains animaux peuvent être des compagnons. Ils peuvent alors *jouer*. Cette méthode ne fera rien, à part afficher un texte décrivant le jeu de l'animal. Parmi les animaux, nous voudrions que **seuls** les `ChienBergerBlanc` et les `Chat` aient cette caractéristique.

(Q3.1) Ajoutez une méthode `jouer` dans les classes `ChienBergerBlanc` et `Chat`.

(Q3.2) Dans `CentreSPATab`, écrivez les deux méthodes suivantes :

— *faireJouer* : fait jouer tous les compagnons du centre.

— `getCompagnons` : renvoie la liste des compagnons en résultat.

(Q3.3) Téléchargez l'interface `Compagnon` fournie sur Moodle et utilisez la pour simplifier votre code.

Grandes lignes :
— Faites en sorte que `ChienBergerBlanc` et `Chat` implémentent l'interface Compagnon.
— Exploitez cette notion pour simplifier vos méthodes *faireJouer* et *getCompagnons*.

4 Implémentation d'un Centre SPA via une `LinkedList` (60mn)

Dans cette version, on n'essaie plus de maintenir une bijection entre les cases du tableau et les cages/box du centre. On souhaite juste que la caractéristique d'un centre soit la liste des animaux présents actuellement.

(Q4.1) Créez une nouvelle implémentation du centre SPA (`CentreSPALinkedList`), en utilisant une `LinkedList` comme moyen de stockage. Vous aurez besoin d'importer : import java.util.LinkedList; et de consulter la documentation.

(Q4.2) Rajoutez la méthode `public LinkedList<Animal> getAnimaux()` qui permet d'obtenir l'ensemble des animaux présents dans le centre sous forme de liste.

Note : dans la vraie vie, on utiliserait la méthode `clone()`, qui permet d'obtenir la copie d'une liste.

4.1 Bonus - s'il reste du temps en séance 2

Faites en sorte que votre implémentation du centre SPA avec un tableau (`CentreSPATab`) dispose de la méthode `getAnimaux()`, avec la même signature.