**JAVA - Basics**

**Atelier TP N° 4**

Héritage - Interfaces

**Exercice 1 :**

On veut modéliser une application pour effectuer l'inventaire d'une bibliothèque. Elle devra traiter : des livres et des dictionnaires. Tous les documents possèdent un numéro d’enregistrement et un titre.

A chaque livre est associé, en plus, un auteur et un nombre de pages, les dictionnaires possèdent une langue et un nombre de tomes. On veut manipuler tous les articles de la bibliothèque au travers de la même classe : **Document**.

1. Définir les classes **Document**, **Livre** et **Dictionnaire**.

2. Définir la classe **ListeDeDocuments** permettant de créer une liste vide de documents, implémenter une méthode permettant d'ajouter un document.

3. Dans la classe **ListeDeDocuments** définir une méthode **tousLesAuteurs**() qui affichera la liste des numéros des documents de la liste avec, pour chacun, l'éventuel auteur.

4. Redéfinir la méthode **toString**() dans Document ainsi que dans les classes Livre et Dictionnaire et qui renvoie une chaîne de caractères décrivant un document, un livre ou un dictionnaire. Ajoutez alors dans la classe **ListeDeDocuments** une méthode **tousLesDocuments**() qui affiche le contenu de tous les documents.

**Exercice 2 :**

La classe **Local** contient les attributs suivants : **proprio**(String), **adresse** (String) et **surface** (double). Les méthodes :

**double** **impot**() : calculer et retourner l’impôt à payer, 5 € par m2 de surface.

**void** **affiche**() : affiche tous les attributs d’un objet **Local**.

*1 – Donner la classe* **Local***.*

Un **Local** peut être un domicile ou professionnel. L’impôt sur une maison est calculé en fonction de la surface, le nombre de pièces (200 €/pièce) ainsi que l’existence ou pas d’une piscine (1000 € supplémentaire). L’impôt sur un local professionnel est calculé en fonction de la surface ainsi que le nombre d’employés (2000 € supplémentaire par tranche de 10 employés)

*2 – Ajouter les classes* **LocalIndiv** *et* **LocalProf***, prévoir les rédéfinitions nécessaire.*

On désire enfin calculer l’impôt au niveau de toute une commune, utiliser un tableau d’objets.

*3 – Simuler le fonctionnement dans une fonction main.*

**Exercice 3 :**

La classe Eleve possède trois attributs privés :

* nom, de type String,
* un ensemble de notes, listeNotes, qui sont des entiers rangés dans un ArrayList<Integer>
* moyenne, égale à la moyenne des notes contenues dans l'attribut listeNotes. Un élève sans aucune note sera considéré comme ayant une moyenne nulle.

La classe Eleve possède :

Un constructeur permettant uniquement d'initialiser le nom de l'élève.

Cinq méthodes publiques :

* Un getter pour la moyenne de l'élève, renvoie la valeur de l'attribut moyenne ;
* Un getter pour le nom de l'élève, renvoie le nom de l'élève ;
* Un getter pour la liste des notes de l'élève, renvoie la liste des notes de l'élève ;
* La méthode ajouterNote(**int** note, ajoute la note reçue en paramètre à listeNotes ; si la note reçue en paramètre est négative, la note introduite est 0 ; si la note reçue en paramètre est supérieure à 20, la note introduite est 20 ; la méthode actualise en conséquence l'attribut moyenne.
* La méthode **public String** toString(), retourne une description de l'élève courant nom et moyenne (par exemple : "Ali (12)").

**Exercice 4 :**

Créez une classe « **House** », avec un attribut « **surface** », un constructeur qui définit sa valeur et une méthode « **Display** » pour afficher « Je suis une maison, ma surface est de XXX m2 » (XXX: la valeur de surface). Incluez aussi des getters et des setters pour la surface.

La classe « **House** » contiendra une porte (**Door**). Chaque porte aura un attribut « **color** » (de type String), et une méthode « **Display** » qui affichera « Je suis une porte, ma couleur est bleu » (ou quelle que soit la couleur). Inclure un getter et un setter. Créez également la méthode « **GetDoor** » dans la classe « **House** ».

La classe « **Apartment** » est une sous-classe de la classe « **House** », avec une surface prédéfinie de 50m2.

Créez également une classe **Person**, avec un **nom** (de type String). Chaque personne aura une maison. La méthode « **Display** » pour une personne affichera son nom, les données de sa maison et les données de la porte de cette maison.

Écrivez un **main** pour créer un **Apartment**, une personne pour y vivre et pour afficher les données de la personne.

**Exercice 5 :**

La classe Eleve implémente l'interface [java.lang.Comparable<T>](http://download.oracle.com/javase/6/docs/api/java/lang/Comparable.html). C'est une interface générique, comme l'indique le <T>. Il faudra, en implémentant l'interface, indiquer à la place du T le type des objets qu'on veut comparer entre eux.

L'interface Comparable<T> déclare une seule méthode :   
    **public int** compareTo(T o);   
Quand cette méthode est implémentée, elle doit retourner une valeur strictement négative, nulle ou strictement positive selon que l'objet concerné (celui qui correspond à la référence **this**) est plus petit que l'objet o, égal à l'objet o ou plus grand que l'objet o.

On comparera les élèves selon leur moyenne.  
Compléter l'en-tête de la classe Eleve par "**implements** Comparable<Eleve>". Il faudra de plus définir la méthode compareTo déclarée par l'interface Comparable. Cette méthode est à nouveau générique ; si la classe implémente Comparable<Eleve>, le paramètre de la méthode doit être de type Eleve, l'en-tête sera donc :   
    **public int** compareTo(Eleve e)