Université Paris Diderot **Programmation Orientée Objet**
L3 **Année 2014-2015**

TP n°3

Introduction à l'héritage

On veut modéliser une application devant servir à gérer l'inventaire d'une médiathèque. La classe `Mediatheque` contiendra la méthode principale `main` permettant de tester les différentes classes de ce TP.

Notes sur l'héritage :
Le but du TP est d'apprendre à se servir de l'héritage en Java. L'héritage permet, lorsque l'on définit une classe, de réutiliser une partie du code et des attributs de la classe parente à l'aide du mot clef `super`. Par exemple, si l'on souhaite définir des classes `Vehicule` et `Voiture`, on pourra écrire :

```
public class Vehicule {
    private int nombreDeRoues;
    public Vehicule(int n) {
        this.nombreDeRoues = n;
    }
    public String toString() {
        return "Ce vehicule a " + this.nombreDeRoues + " roues.";
    }
}

public class Voiture extends Vehicule {
    private String couleur;
    public Voiture(String couleur) {
        super(4);   // Appelle le constructeur parent
        this.couleur = couleur;
    }
    public String toString() {
        String s = super.toString();    // Appelle la méthode parente
        return s + "\n" + "Et c'est une voiture " + this.couleur + " !";
    }
    public static void main(String[] args) { // Permet de tester le code
        Vehicule v = new Voiture("rouge");
        System.out.println(v.toString());
    }
}
```

Attention : l'héritage demande de bien réfléchir à l'organisation des différentes classes. Par exemple, comment devrait-on s'y prendre pour définir une classe `Velo` ? Et une classe `VoitureVolante` ?

Exercice 1 Une médiathèque contient différents types de médias. Mais quelque soit le média, celui-ci possède un *titre*. Quand un média est créé, son titre est donné à la création et par la suite, il ne change plus.

1. Définissez la classe `Media` avec son constructeur public, un champ `titre` privé et son accesseur. Quel attribut devez-vous rajouter pour interdire les modifications du `titre` ?
2. On veut attribuer un *numéro d'enregistrement* unique dès que l'on crée un objet `Media` : le premier média créé doit avoir le numéro 0, puis ce numéro s'incrémente de 1 à chaque création de média. Ajoutez les champs nécessaires à la classe `Media`, modifiez le constructeur puis ajoutez une méthode `getNumero` renvoyant le numéro d'enregistrement du média. Quel(s) attribut(s) faut-il rajouter pour s'assurer que le numéro d'enregistrement restera unique et non-modifiable ?
3. Définissez la méthode `toString` renvoyant la chaîne de caractères constituée du numéro d'enregistrement et du titre du média.
4. Définissez la méthode `plusPetit(Media doc)` qui vérifie que le numéro d'enregistrement de l'instance courante est plus petit que celui de *doc*.

Exercice 2 On veut maintenant définir des médias de natures diverses : des livres, des dictionnaires, et, plus tard, d'autres types de médias (bandes dessinées, dictionnaires bilingues,...). A chaque livre sont associés, en plus, un *auteur* et un *nombre de pages*. Les dictionnaires ont, eux, pour attributs supplémentaires une *langue* et un *nombre de tomes*. On voudrait pouvoir manipuler tous les articles (livres, dictionnaires, etc.) au travers de la même représentation : celle de média.

1. Définissez les classes `Livre` et `Dictionnaire` étendant la classe `Media`. Définissez pour chacune un constructeur permettant d'initialiser toutes ses variables d'instances respectives.
2. Redéfinissez la méthode `toString()` dans les classes `Livre` et `Dictionnaire` pour qu'elle renvoie une chaîne de caractères décrivant un livre ou un dictionnaire, en plus de la description normale d'un média.
3. Définissez ensuite quelques classes supplémentaires, par exemple `BandeDessinee`, `Manga`, `DictionnaireBilingue`, etc. avec des propriétés en plus. Pour chacune de ces nouvelles classes, quelle classe étend-elle ?
4. Rajoutez une méthode `main` pour tester votre code : par exemple, dans la classe `Livre`, créez successivement deux livres « Harry Potter » (écrit par J. K. Rowling, faisant 240 pages) et « Le Trône de fer » (écrit par George R.R. Martin, faisant 789 pages). Affichez ces livres, et assurez-vous que le numéro d'enregistrement est le bon.

Rappel (structures de données) :
De nombreuses structures de données sont disponibles en Java, et peuvent-être utilisées avec tout type d'objets (par exemple des objets de type `Media`). Pour cela, on peut utiliser des classes *génériques* comme la classe `LinkedList`. Pour préciser à Java que l'on crée une crée une `LinkedList` qui contiendra des `Media`, on définit et on initialise la liste en écrivant
`LinkedList<Media> maListe = new LinkedList<Media>();`.

Exercice 3 On veut maintenant définir une médiathèque pour ranger tous nos médias. Il s'agit d'une classe contenant l'attribut `LinkedList<Media> baseDeDonnees`, des méthodes opérant sur celle-ci et un main pour faire les tests de ce TP.

1. Définissez la classe `Mediatheque` ainsi qu'un constructeur affectant une base de donnée vide, et une méthode `toString` décrivant le contenu de la médiathèque.
2. Définissez une méthode `ajouter(Media doc)` qui ajoute *doc* dans la base de donnée toute en la laissant triée pour l'ordre `plusPetit`.

3. Testez tout ceci dans le `main` de la classe `Mediatheque` : créez une liste contenant plusieurs médias (de différents types) puis affichez-la.
4. Pourrait-on définir une classe `Bibliotheque` qui étend `Mediatheque` en ne contenant que des objets de type `Livre` ? Est-ce une bonne idée ?

Exercice 4 (Avancé — pour les L3) Le logiciel développé pour la médiathèque doit également être capable de gérer les emprunts. On veut donc ici définir des classes représentant les abonnés de la médiathèque (qui bénéficient de différents abonnements), et ajouter les bonnes méthodes pour faire interagir ces différentes classes.

1. Les abonnés peuvent, par défaut, emprunter un seul média à la fois. Les étudiants, eux, peuvent en emprunter deux simultanément. Définissez une classe `Abonné`, comprenant le nom et le prénom de l'abonné, ainsi qu'une sous-classe `EtudiantAbonné`. Ajoutez l'attribut privé `empruntsMax`.
2. Quelle(s) classe(s) faut-il modifier pour pouvoir emprunter des médias ? Attention : deux abonnés ne peuvent pas emprunter le même média en même temps dans une même médiathèque. Définissez des méthodes `emprunter` et `rendre`, permettant d'emprunter ou de rendre des médias à une médiathèque. En cas d'erreur, on se contentera d'afficher un message à l'utilisateur. Pouvez-vous profiter de l'héritage pour réutiliser du code (c'est-à-dire ne pas écrire le même code deux fois) ?
3. Selon les choix que vous avez faits, modifiez les autres classes pour pouvoir emprunter des médias. Écrivez un programme permettant de tester toutes vos classes (15 lignes environ).

Rappel (opérateur `instanceof`) :
L'expression `x instanceof UneClasse` vaut `true` si `x` est un objet de la classe `UneClasse` ou d'une classe héritière.

Exercice 5 (Avancé — pour les L3) Lorsque l'on affiche la base de données de la médiathèque les médias sont classés par numéro d'enregistrement, mélangeant des médias de types différents. Ce n'est pas très lisible. Nous aimerions modifier l'ordre pour que les média d'un même type apparaissent à côté. Nous essayons ici de séparer les livres du reste.

1. Redéfinissez la méthode `plusPetit(Media doc)` de la classe `Livre` de sorte que l'instance courante soit considérée comme plus grande que `doc` si celui-ci n'est pas un livre (on utilisera `instanceof`). Lorsque `doc` est un livre on réutilisera la version précédente de `plusPetit`
2. Que cherche-t-on ainsi à faire ? Testez sur plusieurs exemples. Obtient-on le résultat recherché ? Pourquoi ?
3. Ajouter dans la classe `Media` une méthode `plusPetit(Livre doc)` de sorte que `doc` soit considéré comme plus grande que l'instance courante si celle-ci n'est pas un livre (essayez d'utiliser la version de `plusPetit(Media doc)` de la classe `Livre`).
4. Testez le programme sur plusieurs exemples. Celui-ci devrait boucler. Pourquoi ? Corrigez le bug et vérifiez que tout marche.
5. Cette version n'est pas satisfaisante, pourquoi ?