Université Paris Diderot — Programmation Orientée Objet
Licence Informatique — Année 2009–2010

TP n°4 - Correction

Révisions, suite et fin. Introduction à l'héritage

On modélise une application devant servir à l'inventaire d'une bibliothèque. Elle devra traiter des documents de nature diverse : des livres, des dictionnaires, et autres types de documents qu'on ne connaît pas encore précisément mais qu'il faudra certainement ajouter un jour (bandes dessinées, dictionnaires bilingues,...).

On définira une classe `Bibliotheque` réduite à la seule méthode `main` pour tester les différentes classes de ce TP.

Exercice 1 Tous les documents possèdent un *titre*. Quand un document est créé, son titre est donné à la création et par la suite, il ne change plus.

1. Définir la classe `Document` avec son constructeur public, et la propriéte `titre` privée et son accesseur.
2. On veut attribuer un *numéro d'enregistrement* unique dès que l'on crée un objet `Document`. On veut que le premier document créé ait le numéro 0, et que ce numéro s'incrémente de 1 à chaque création de document. Quelles propriétés faut-il ajouter à la classe `Document` ? Lesquelles doivent-être `static` ? Les ajouter sans leurs accesseurs, et modifier le constructeur. Puis ajouter une méthode `getNumero` renvoyant le numéro d'enregistrement du document.
3. Ajouter un second constructeur public permettant de définir aussi, en plus du titre, le numéro d'enregistrement (supposé positif ou nul) lors de la création d'un document. Quelles restrictions peut-on prendre afin d'éviter que deux documents créés aient le même numéro ? Dans les cas de restriction, on prendra le numéro suivant prévu, à la place du numéro indiqué par l'utilisateur.
4. Redéfinir la méthode `toString` renvoyant la chaîne de caractères constituée du numéro d'enregistrement et du titre du document.

Correction : Il faut prendre garde à ce que, si on ajoute un document en indiquant son numéro, ce numéro soit plus grand que le numéro suivant prévu. Il faut de plus mettre à jour le numéro suivant.

```
public class Document {
    static private int numeroSuivant = 0;
    private int numero;
    private String titre;

    public Document(String titre) {
        this.numero = numeroSuivant;
        numeroSuivant++;
        this.titre = titre;
    }

    public Document(String titre, int numero) {
        if (numero < numeroSuivant) {
           numero = numeroSuivant;
        }
```

```
        this.numero = numero;
        numeroSuivant = numero + 1;
        this.titre = titre;
    }

    public int getNumero(){
        return numero;
    }

    public String getTitre(){
        return titre;
    }

    public String toString() {
        return ("numero: "+numero+" titre: "+titre);
    }
}
```

Exercice 2 On veut implémenter une liste de documents dans laquelle on veut pouvoir ajouter des documents petit à petit, et y accéder par leur numéro d'enregistrement. Mais plutôt que d'utiliser les structures de collections Java (`ArrayList`, `Vector`, etc. que l'on verra dans une séance ultérieure), on veut programmer soi-même "en dur" une structure de données.

Pour implémenter cette structure, on va créer une classe `ListeDeDocuments` avec un tableau de documents `tableau` contenant les documents de la liste. Cependant, pour éviter d'avoir à augmenter la taille du tableau à chaque ajout, on va fixer initialement sa taille à 1 et doubler sa taille dès qu'on veut ajouter un document dont le numéro d'enregistrement est plus grand que la taille du tableau.

1. Définir la classe `ListeDeDocuments` avec la propriété privée `tableau` et la méthode `ajouter` qui ajoute un document à la place indiquée par son numéro d'enregistrement, ainsi qu'une méthode `getDocument` qui prend en argument un numéro et renvoie le document de la liste ayant ce numéro (`null` s'il n'est pas dans la liste).
2. Dans cette classe, redéfinir la méthode `toString` pour qu'elle renvoie les descriptions de tous les documents de la liste, séparées par un saut de ligne `"\n"`.
3. Modifier la classe `Document` pour faire en sorte que tous les documents créés puissent être retrouvés en utilisant une méthode statique `accesDocument` dans la classe `Document`.

Correction :

```
public class ListeDeDocuments {

    private Document[] tableau;

    public void ajouter(Document doc) {
        if (tableau.length <= doc.getNumero()) {
            Document[] t = new Document[tableau.length * 2];
            for (int i = 0; i < tableau.length; ++i) {
              t[i] = tableau[i];
            }
            tableau = t;
        }
```

```
        tableau[doc.getNumero()] = doc;
    }

    public Document getDocument(int index) {
        if (index >= tableau.length) {
            return null;
        }
        return tableau[index];
    }

    public ListeDeDocuments() {
        tableau = new Document[1];
    }

    public String toString() {
        String resultat = "";
        for (int i = 0; i < tableau.length; ++i) {
            Document d = tableau[i];
            if (d != null) {
            resultat += d + "\n";
            }
        }
        return resultat;
    }
}
```

L'inconvénient de cette représentation est qu'il est difficile de parcourir la liste des documents. Cela est dû au fait que cette liste, optimisée pour l'accès direct aux documents par leur numéro, peut ainsi comporter des "trous".

Exercice 3 La bibliothèque est appelée à traiter des documents de nature diverse : des livres, des dictionnaires, et autres types de documents.

À chaque livre est associé, en plus, un *auteur* et un *nombre de pages*, les dictionnaires ont, eux, pour attributs supplémentaires une *langue* et un *nombre de tomes*. On veut manipuler tous les articles de la bibliothèque au travers de la même représentation : celle de document.

1. Définissez les classes `Livre` et `Dictionnaire` étendant la classe `Document`. 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 ununn livre ou un dictionnaire, en plus de la description normale d'un document.

   Correction :

   ```
   public class Livre extends Document{
       private String auteur;
       private int nbPages;

       public Livre(String titre, String auteur, int nbPages) {
           super(titre);
           this.nbPages = nbPages;
           this.auteur = auteur;
       }
   ```

```
    public String getAuteur() { return auteur; }
    public int getNombreDePages() { return nbPages; }
    public String toString(){
        return (super.toString()+" auteur: "+getAuteur()+" pages: "+getNombreDePages());
    }

}

public class Dictionnaire extends Document {
    private int nbTomes;
    private String langue;

    public Dictionnaire(String titre, int nbTomes, String langue) {
        super(titre);
        this.nbTomes = nbTomes;
        this.langue = langue;
    }
    public int getNombreDeTomes() { return nbTomes; }
    public String getLangue() { return langue; }
    public String toString() {
        return (super.toString()+" tomes: "+getNombreDeTomes()+" langue: "+getLangue());
    }
}
```

3. Dans la classe `ListeDeDocuments`, définissez une méthode `tousLesAuteurs` qui renvoie la chaîne des auteurs de tous les livres de la liste.
4. Dans la classe `ListeDeDocuments` définissez une méthode `tousLesDicos` qui renvoie une liste de documents (sous la forme d'une nouvelle instance de la classe `ListeDeDocuments`) contenant tous les dictionnaires de la liste.
   Correction :

```
    public ListeDeDocuments tousLesDicos() {
        ListeDeDocuments resultat = new ListeDeDocuments();
        for (int i = 0; i < tableau.length; ++i) {
            Document d = tableau[i];
            if (d != null) {
                if (d instanceof Dictionnaire) {
                    resultat.ajouter(d);
                }
            }
        }
        return resultat;
    }

    public String tousLesAuteurs() {
        String resultat = "";
        for (int i = 0; i < tableau.length; ++i) {
            Document d = tableau[i];
            if (d != null) {
                if (d instanceof Livre) {
                    resultat += ((Livre) d).getAuteur() + "\n";
                }
            }
        }
```

```
        return resultat;
    }
```

5. 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 ?

Exercice 4 Dans l'exercice suivant nous allons utiliser la classe `FileWriter` pour générer un fichier. Pour l'utiliser il faut l'importer `import java.io.FileWriter;`. D'autre part, l'utilisation des constructeurs de cette classe, peuvent générer des exceptions du type `IOException`. Il convient donc :

- d'importer la classe `IOException` avec `import java.io.IOException;`,
- d'ajouter `throws IOException` à la déclaration des méthodes dans lesquelles nous utilisons `FileWriter`.

Un cours et un TD seront consacrés à la gestion des exceptions.

Enfin différents constructeurs existent pour `FileWriter`. Nous allons utiliser le constructeur prenant en argument une chaine de caractères. Notez que cette chaîne de caractères peut spécifier le chemin vers le dossier dans lequel nous voulons générer nos fichiers. Par exemple :

```
FileWriter w = new FileWriter(‘‘/local/home/monom/file’’);
```

génère un fichier `file.txt` vide dans mon dossier `/local/home/monom/`.

Pour ecrire dans un tel fichier, on utilise entre autres les méthodes :

- `write(String s)` qui ecrit la chaîne s dans le fichier,
- `close()` pour mettre fin à l'écriture dans ce fichier.

Si vous utilisez `write(String s)`, il est nécessaire de terminer l'ecriture du fichier en utilisant `close()`.

- Dans la classe `ListeDeDocuments`, définissez une fonction `sauvegarde` permettant d'écrire la liste des documents dans un fichier.

Correction :

```
public void sauvegarde(String filename) throws IOException {
  FileWriter w = new FileWriter(filename);
  Document unDocument;
  for (int i=0; i<lesDocuments.size(); i++){
    unDocument = lesDocuments.elementAt(i);
    w.write(unDocument.toString()+ "\n");
  }
  w.close();
}
```