Examen décembre 2009

3 heures - documents écrits autorisés
livres et calculatrice interdits

▷ La clarté des réponses sera prise en compte dans l'évaluation. Evidemment vos réponses devront être convaincantes et donc précises !

Exercice 1 : Médiathèque

Une médiathèque regroupe un certain nombre de documents qu'il est possible de consulter. Chacun de ces documents est décrit par un certain nombre d'informations le concernant.
Les documents sont de différentes natures. On trouve de la musique et des livres, mais l'intégration d'autres types (films, logiciels, etc.) doit être réalisable.

Auteurs L'auteur d'un document peut être soit une personne unique soit un groupe de personnes dans le cas de plusieurs auteurs. On veut une méthode `toString` pour le type représentant les auteurs. Les informations disponibles sur une personne sont son nom et ses années de naissance et décès. Cette dernière valeur vaudra `-1` si la personne est encore vivante. Ces informations apparaissent dans la méthode `toString` des personnes, pour les groupes d'auteurs, les informations pour chacun des auteurs sont concaténées.

Documents Les documents seront du type `Document`. Un document est caractérisé par un titre, un auteur. Le *titre* sera représenté par une chaîne de caractères.
Il est possible de consulter un document. Les abonnés à la médiathèque peuvent emprunter un document. Il est donc possible de savoir si un document est emprunté ou disponible.
Les documents sont répartis entre les livres et les musiques.

- Les livres sont en plus caractérisés par un nombre de pages et un numéro ISBN (une chaîne de caractères). Consulter un livre c'est le lire.
- Les musiques sont caractérisées par une durée en nombre entier de secondes et un genre musical (parmi *rock*, *pop*, *classique* et *jazz* par exemple). Consulter une musique c'est l'écouter.

On veut également disposer d'une méthode `toString` reprenant les différentes informations sur chacun des documents.
Les données concernant un document sont fixées à la construction de l'objet.

La mediathèque Dans la médiathèque chaque document est identifié par sa référence, unique, générée à partir du document. Les références sont gérées par la classe `Reference` présentée ci-dessous :

Reference
...
- Reference() +genereReference(doc : Document):Reference (static) +toString():String +hashCode():int +equals(o : Object):boolean

La méthode `genereReference` permet d'obtenir la référence du document paramètre;
On doit pouvoir :

- ajouter un document de la médiathèque,
- le supprimer à partir de sa référence

- emprunter et rendre un document à partir de sa référence ; dans le premier cas une exception `DejaEmprunteException` est levée si le document est déjà emprunté,
- accéder à la collection de tous les documents de la médiathèque.
- On définit le paquetage `mediatheque` comme racine de ce projet. Le cas échéant, vous préciserez les paquetages que vous définissez.
- Dans la suite, dans les diagrammes UML, vous ferez apparaître les méthodes (signature complète) et attributs ainsi que les éventuelles redéfinitions/surcharges.

Q 1 . Donnez le(s) diagramme(s) UML des types permettant de gérer les auteurs.

Q 2 . Donnez sous forme de diagrammes UML la hiérarchie de types qui vous parait la plus appropriée pour définir les types (classes ou interfaces) nécessaires à la modélisation des documents décrits ci-dessus.

Q 3 . Donnez **tout** le code (classe et super-types, sauf `Object`) nécessaire pour la modélisation des documents de type musique. Le code des types pour les auteurs n'est pas demandé.

Pour la méthode d'écoute d'une musique vous mettrez "`...`" dans le corps de méthode.

Q 4 . Donnez le code java d'une `Mediatheque` (pour simplifier l'exercice, on ne s'occupe pas ici de mémoriser qui emprunte un document).

Q 5 . On souhaite maintenant pouvoir faire évoluer la notion de documents, sans modifier l'existant. On considère ici deux "améliorations", la première consiste à ajouter des mots-clés à un document, la seconde à préciser qu'un document est un "document téléchargeable" (pour les documents électroniques).

D'abord, afin d'améliorer la recherche sur les documents, on souhaite pouvoir ajouter des mots-clés sur un document. On crée ainsi la notion de "`DocumentRiche`" : un `DocumentRiche` est un `Document`. En plus de ce qu'il est possible de faire pour un document, on peut lui ajouter des mots-clés (chaîne de caractères) et savoir s'il satisfait un mot-clé donné m, c'est-à-dire m est l'un des mots-clés qui caractérisent le *DocumentRiche*. La description par `toString` d'un document riche reprend la description du document enrichi et précise en plus les mots-clés qui ont été attachés à ce document.

Ensuite, on souhaite prendre en compte que certains documents sont en fait des documents électroniques qui peuvent donc être téléchargés. On ajoute la notion de `DocumentTelechargeable` : un `DocumentTelechargeable` est un `Document`. Il dispose en plus d'une information qui précise l'URL à laquelle on peut récupérer ce document, cette information est du type `java.net.URL`. Evidemment un document électronique n'est jamais considéré comme emprunté puiqu'il peut être téléchargé autant de fois que nécessaire. La description d'un document téléchargeable précise son URL en plus de la description du document.

Bien évidemment, on peut avoir des documents (livres ou musiques) riches qui sont téléchargeables ou enrichir des documents téléchargeables.

Q 5.1. Donnez le diagramme UML de votre proposition pour prendre en compte ces 2 nouveaux types de document, vous indiquerez clairement le lien avec le diagramme de votre réponse à la question 2.

Q 5.2. Donnez le code JAVA correspondant à ces types.

Q 5.3. Indiquez comment, à partir d'un document `doc` existant, créer un document riche téléchargeable dont les mots clés sont `"mot1"` et `"mot2"` et dont l'URL est l'objet de référence `monUrl`.

Sélections et abonnements.

Que ce soit pour permettre une recherche parmi l'ensemble des documents ou pour gérer différents types d'abonnements à la médiathèque, il est nécessaire de disposer d'outils permettant de sélectionner des documents en fonction de critères.
On définit le type `DocumentSelectionneur` comme disposant au minimum (il sera complété un peu plus loin) des méthodes :

- `public Collection<Document> selectionne(Collection<? extends Document> docsInitiaux)` qui retourne la collection des documents acceptés par le sélectionneur parmi les documents de `docsInitiaux`.
- `public boolean estAccessible(Document doc)` qui vaut `true` ssi `doc` est un document accepté par ce sélectionneur.

On doit ainsi pouvoir envisager des sélectionneurs ne fournissant que les livres, ou que les musiques, ou que les livres dont le titre contient un certain mot, etc.

Abonnement Un abonnement est ce qui permet à un utilisateur de la médiathèque d'emprunter un livre. L'abonnement définit en particulier les documents qu'il permet d'emprunter.
Ainsi il existe des abonnements qui ne permettent de n'emprunter que les livres, d'autres que de la musique jazz, que des documents téléchargeables, etc.
La restriction des documents accessibles à un abonnement est réalisée à l'aide d'un sélectionneur tel que défini ci-dessus.

Q 6 . Compléter le code ci-dessous aux trois endroits indiqués :

```
1  package mediatheque.abonnement;
2  COMPLETER_1
3  public class Abonnement {
4      protected Mediatheque mediatheque;
5      protected DocumentSelectionneur selectionneur;
6      /** construit un abonnement
7       * @param mediatheque la médiathèque pour laquelle cet abonnement est actif
8       * @param selectionneur le sélectionneur qui permet de filtrer les documents
9       * de la médiathèque  accessibles pour cet abonnement
10      */
11     public Abonnement(Mediatheque mediatheque, DocumentSelectionneur selectionneur) {
12         this.mediatheque = mediatheque;
13         this.selectionneur = selectionneur;
14     }
15
16     public Mediatheque getMediatheque() {
17         return mediatheque;
18     }
19     /** fournit les documents de la mediathèque accessibles pour cet abonnement
20      * @return les documents accessibles
21      */
22     public Collection<Document> documentsAccessibles() {
23         COMPLETER_2
24     }
25    /** permet l'emprunt d'un document par cet abonnement
26     * @param ref la réference du document à emprunter
27     * @throws DocumentHorsAbonnementException si le document n'est pas accessible
28     *     pour cet abonnement
29     * @throws DejaEmprunteException si le document est déjà emprunté
30     */
31     public void emprunte(Reference ref) throws DocumentHorsAbonnementException,
32                                                DejaEmprunteException {
33       COMPLETER_3
34     }
35 }
```

Selectionneur On s'intéresse désormais au type `DocumentSelectionneur`.
Il se pose en particulier le problème de gérer les différents types de documents avec un seul sélectionneur. Ainsi, si l'on veut sélectionner dans la médiathèque les musiques de type *pop* il faut pouvoir accéder à cette information, alors que celle-ci n'existe que pour les documents musicaux, elle n'a aucun sens pour les livres par exemple.

Q 7 . Un `DocumentSelectionneur` est un visiteur pour les éléments de type `Document` via la méthode `estAccessible` qui "interroge" un document afin de savoir si celui-ci accepte ou non d'être visité par ce sélectionneur.

En vous inspirant de la mise en œuvre du design pattern *Visiteur* (cf. Figure 1 pour mémoire), proposez une solution pour mettre en place le type `DocumentSelectionneur`.

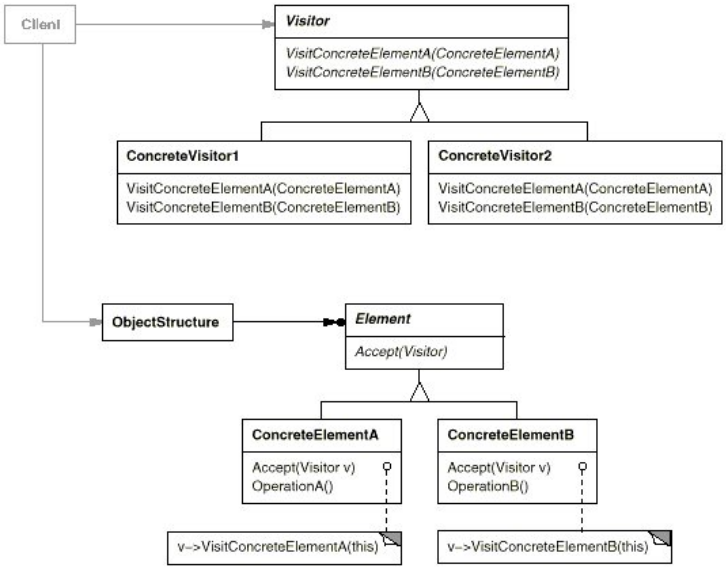

Figure 1: Le design pattern *Visitor*

Vous serez probablement amené à modifier légèrement les types de documents précédemment définis.

Q 7.1. Donnez un diagramme UML détaillé pour le type `DocumentSelectionneur` puis son code Java.

Q 7.2. Indiquez clairement ce qu'il faut modifier dans les types de documents, en indiquant le code à ajouter à la classe des documents musicaux que vous avez définie à la question 3.

Q 7.3. Donnez le code d'une classe `SelectionneurMusique` qui permet de rendre accessible, et donc sélectionner, uniquement les documents musicaux.

Q 7.4. Même question pour un sélectionneur qui rend accessible tous les documents dont le titre contient une chaîne de caractères passées à la construction du sélectionneur.

Q 7.5. On définit un sélectionneur composé comme un sélectionneur réalisant l'union de plusieurs autres sélectionneurs. Ainsi un sélectionneur composé des sélectionneurs s_1 et s_2 rendra accessible les documents accessibles par s_1 ou par s_2.

Donnez un code pour une classe permettant de définir des sélectionneurs composés.