Conception Orientée Objet

Examen

3 heures - documents de cours autorisés, livres interdits
mercredi 9 juin 2004

Les deux exercices sont indépendants. L'ordre des exercices ne présage pas de leurs difficultés respectives.

Exercice 1 : Documents : contenus et formatages

On travaille sur des documents dont la structure correspond à celle du document présenté en annexe 1.3. Nous nous intéresserons dans un premier temps à la représentation du contenu des documents et dans un second à leur visaulisation. Les deux parties peuvent être traitées indépendamment.

Q 1 . Donnez le code des classes dont les diagrammes sont donnés à la figure 2, mise à part la classe `TextPartIterator` dont le code est donné en annexe et vous ne donnerez le code que pour l'une des deux classes `Subsection` ou `Section`.

Une même contenu de document peut être visualisé de différentes manières. La visualisation d'un document nécessite généralement le formatage de son contenu, le format étant ensuite interprété par le logiciel de visualisation. Vous trouverez en annexe deux formats d'un même document, le premier est HTML, le second est LaTeX. Un rapide examen permet de trouver des points communs dans la construction de ces formats de sortie. En effet, chacun des formats :

- commence par une entête définissant un certain nombre de propriétés propres au format,
- traite le contenu effectif du document (voir ci-dessous),
- dispose d'une conclusion (`</html>` dans un cas, `\end{document}` dans l'autre).

Le traitement du contenu du document suit lui aussi une démarche ordonnée commune :

- traitement du titre,
- traitement du résumé puis des mots-clés,
- traitement séquentiel des sections, qui se décompose en :
  - gestion du titre,
  - gestion séquentielle des sous-sections, qui se décompose en :
    - gestion du titre,
    - gestion séquentielle des paragraphes.

On peut également remarquer que la construction des titres des différents niveaux est similaire d'un niveau à l'autre, au nommage des niveaux près.

Q 2 . En utilisant **au mieux** l'héritage, faites une proposition orientée objet permettant de formater des documents selon cette procédure. Vous l'appliquerez à la création des formats HTML et LaTeX comme dans les exemples.

Il doit être possible d'ajouter un nouveau type de formatage respectant cette structure (en XML par exemple) sans devoir modifier le code existant.

Donnez le code JAVA correspondant à votre solution. Votre proposition devra au moins comporter une méthode ayant la signature `public void formater(Document doc)` qui réalisera le formatage du document en paramètre selon le format voulu. Le format construit sera affiché sur la sortie standard.

Ainsi, le formatage du document présenté à l'annexe 1.3, représenté par les objets conformément au diagramme 2, produit, selon le format choisi, les documents des annexes 1.4 ou 1.5.

Remarque : Il n'est jamais demandé dans cet exercice d'écrire une partie de code créant un objet document.

Exercice 2 : Circuits.

On souhaite modéliser des *circuits* électriques construits à partir de composants (voir figure 1). Un composant a au plus deux entrées et au plus 1 sortie. Pour construire un circuit les sorties de composants sont connectées aux entrées d'autres composants. La sortie d'un composant peut être active ou inactive. Cela dépend du composant et des valeurs de ses entrées. On dit qu'une entrée est active (resp. inactive) si elle est reliée à la sortie active (resp. inactive) d'un composant (donc si elle n'est reliée à aucun composant l'entrée est inactive).

Les composants *générateur* n'ont pas d'entrée et ont 1 sortie constamment active.

Les composants *ampoule* ont 1 entrée et pas de sortie. Une ampoule est allumée si son entrée est active.

Les composants *porte* NON et *interrupteur* sont des composants à 1 entrée et 1 sortie. Un *interrupteur* peut être ouvert ou fermé. La sortie d'une *porte* NON est active si son entrée est inactive et réciproquement. La sortie d'un *interrupteur* est active si et seulement si il est fermé et que son entrée est active.

On trouve ensuite d'autres composants à 2 entrées et 1 sortie, telles que les *portes* ET et *les portes* OU. Ces composants réalisent respectivement l'opération logique ET et OU où l'on considère qu'actif correspond à VRAI et inactif à FAUX.

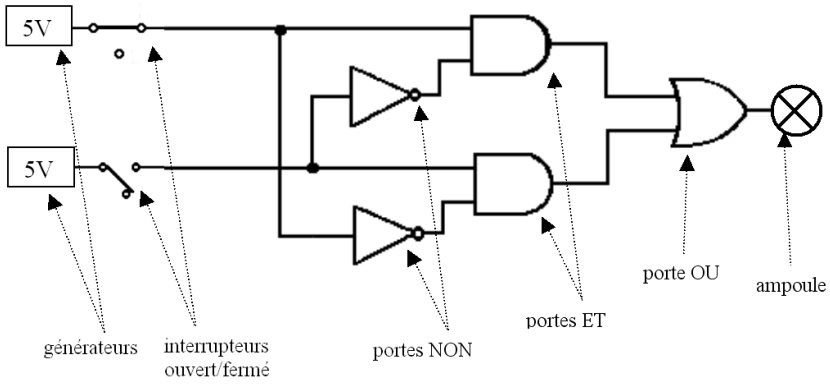

Figure 1: Un exemple de circuit

On dira qu'un composant est actif si sa sortie est active, dans le cas particulier de l'ampoule on dira qu'elle est active si elle est allumée.

L'entrée d'un composant est caractérisée par le composant qui y est connecté. Elle vaut `null` si il n'y en a pas. Il est donc toujours possible de connaître l'état d'activation d'un composant.

Q 1 . Proposez les diagrammes UML des classes qui vous paraissent les mieux adaptées pour modéliser les entités représentant les différents *composants* mentionnés dans le texte précédent. Vous préciserez les types des attributs et des paramètres de méthodes/constructeurs.

Précisez les relations entre ces entités (héritage, implémentation, association).

Q 2 . On peut obtenir de nouveaux composants par combinaison d'autres composants. C'est le cas des portes NON_ET qui correspondent à l'enchaînement d'une porte ET dont la sortie est connectée à l'entrée d'un porte NON ; les entrées de la porte NON_ET sont alors les entrées de la porte ET et sa sortie celle de la porte NON. Il en est de même, par exemple, pour un composant réalisant une porte XOR. D'autres exemples sont possibles.

Q 2.1. Etendez le diagramme UML que vous avez donné à la question précédente pour prendre en compte ces composants obtenus par combinaison. On supposera que ces composants ont toujours 2 entrées et 1 sortie. Vous traiterez en particulier les cas des composants NON_ET et XOR

Q 2.2. Donnez le code JAVA correspondant à votre réponse à la question Q 2.1.

Q 3 . On désire fournir une interface graphique permettant de visualiser les circuits/composants modélisés aux deux précédentes questions, tout en sachant que d'autres composants non encore prévus doivent pouvoir être ajoutés aussi facilement que possible.

On souhaite donc que chaque type de composant dispose d'une icône graphique particulière (comme à la figure 1). Proposez une solution aussi simple que possible, respectant tant que faire se peut le principe ouvert-fermé, pour répondre à ce problème, et indiquez les changements qu'elle implique éventuellement dans vos réponses aux questions précédentes. Quel design pattern avez vous mis en œuvre (la réponse *aucun* est autorisée) ?

1 Annexes

1.1 Le diagramme des classes pour représenter les documents

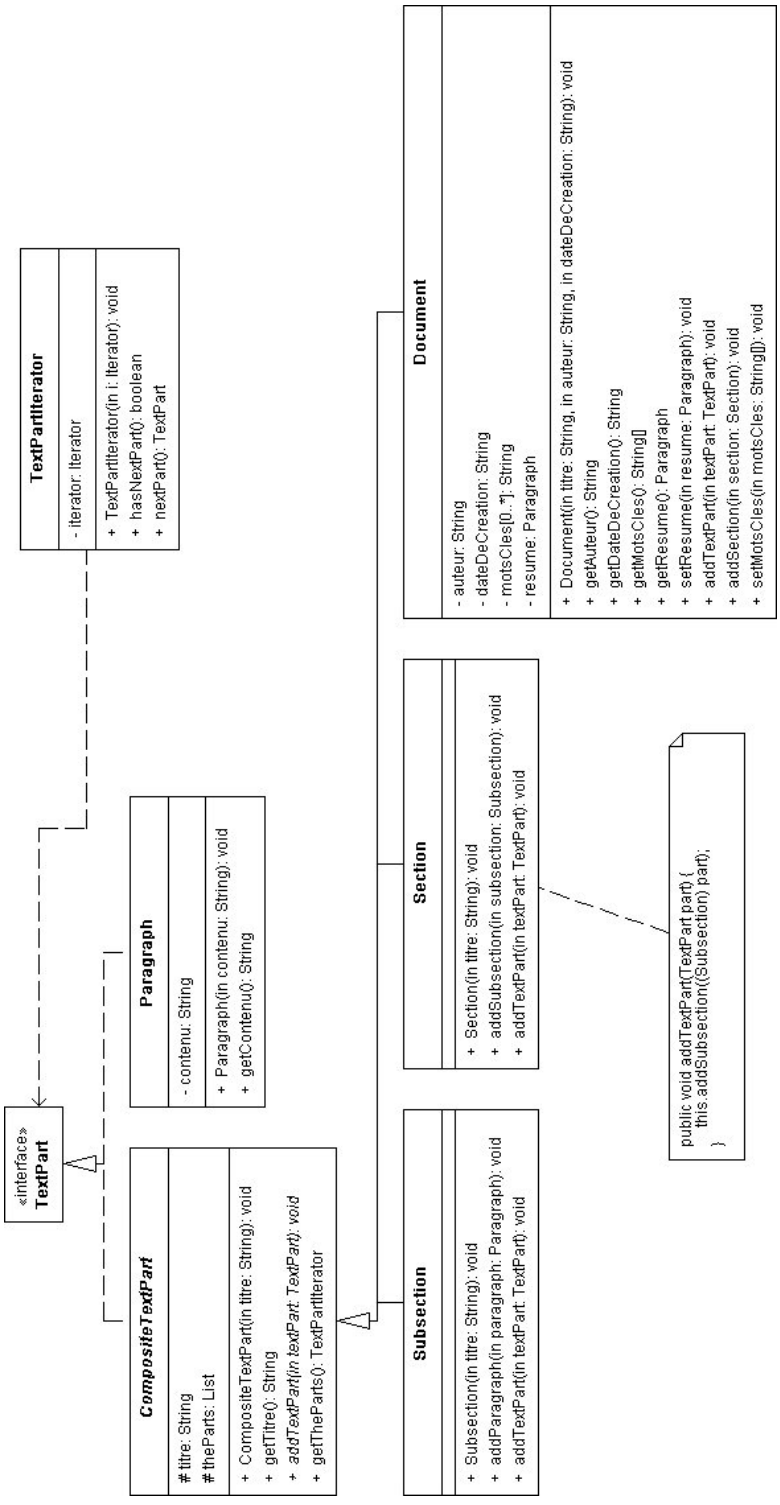

Figure 2: Diagramme UML des classes pour représenter les documents

1.2 La classe `TextPartIterator`

```
package document;
import java.util.*;
public class TextPartIterator {
   private Iterator iterator;
   public TextPartIterator(Iterator i) {
      this.iterator =i;
   }
   public boolean hasNextPart() {
      return iterator.hasNext());
   }
   public TextPart nextPart() throws NoSuchElementException {
      return (TextPart) iterator.next();
   }
}
```

1.3 Le document

Voici un exemple de document structuré. Ce même document est présenté dans les deux annexes suivantes dans deux formats différents reprenant cette structure : le format HTML et le format LaTeX.

titre Titre du document

auteur Auteur du document

date création Date de création

résumé Ceci est un résumé du document...

mots-clés Document, dummy.

sections

1. Première section
   (a) Première sous-section
      - Je suis le premier paragraphe de la première sous-section de la première section.
      - Je suis le second paragraphe de la première sous-section de la première section.

   (b) Seconde sous-section
      - Je suis le premier paragraphe de la seconde sous-section de la première section.
      - Je suis le second paragraphe de la seconde sous-section de la première section.

2. seconde section

1.4 Le document au format HTML

```
<html>
  <header>
    <meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1">
    <title>Titre du document</title>
  </header>

  <body bgcolor=white>
    <p align=center>
      <h1>Titre du document</h1>
      <h3>Auteur du document</h3>
      <h4>Date de création</h4>
    </p>

    <p align=left>
      <i>Résumé</i><br>
      Ceci est un résumé du document...
      <br>
      <b>Mots-clés :</b> Document, dummy.
    </p>

    <h2>Première section</h2>

    <h3>Première sous-section</h3>

    <p> Je suis le premier paragraphe de la première sous-section de
      la première section.
    </p>

    <p> Je suis le second paragraphe de la première sous-section de
      la première section.
    </p>

    <h3>Seconde sous-section</h3>

    <p> Je suis le premier paragraphe de la seconde sous-section de la
      première section.
    </p>

    <p> Je suis le second paragraphe de la seconde sous-section de la
      première section.
    </p>

    <h2>Seconde section</h2>
  </body>
</html>
```

1.5 Le document au format LaTeX

```
\documentclass[a4paper]{article}
\usepackage[latin1]{inputenc}

\title{Titre du document}
\author{Auteur du document}
\date{Date de création}

\begin{document}

\maketitle

\begin{quote}
\begin{center}
\textsl{Résumé}
\end{center}
Ceci est un résumé du document...

\textsc{Mots-clés~:} Document, dummy.
\end{quote}

\section{Première section}

\subsection{Première sous-section}

Je suis le premier paragraphe de la première sous-section de la
première section.

Je suis le second paragraphe de la première sous-section de la
première section.

\subsection{Seconde sous-section}

Je suis le premier paragraphe de la seconde sous-section de la
première section.

Je suis le second paragraphe de la seconde sous-section de la
première section.

\section{Seconde section}

\end{document}
```