Cours de programmation orientée-objet
Examen du 5 juin 2015

Livres fermés. Durée : 3 heures 1/2.

Veuillez répondre à chaque question sur des feuilles séparées sur lesquelles figurent nom, prénom et section. Soyez bref et concis, mais précis.

1. On souhaite développer en Java une classe `BookCatalog` permettant de gérer des catalogues de livres. Une instance de cette classe représente un ensemble (non ordonné) de livres, chacun de ceux-ci étant caractérisé par un titre, une liste d'un ou de plusieurs auteurs (représentés par leur prénom et leur nom), et un commentaire textuel.

   La classe `BookCatalog` doit pouvoir être instanciée sans fournir de paramètre, ce qui crée alors un catalogue vide. Elle doit aussi définir une méthode permettant d'ajouter un livre à un catalogue, et une autre capable d'exporter l'ensemble des données d'un catalogue dans un format choisi par l'utilisateur.

   Afin de standardiser cette opération d'exportation, on impose que la classe `BookCatalog` implémente l'interface suivante :

```
import java.io.IOException;
public interface BookCatalogExporter
{
  void export(String filename, BookCatalogFormat format) throws IOException;
}
```

   Dans cette interface, le paramètre `filename` spécifie le nom du fichier dans lequel la méthode `export` écrit les données exportées. Une erreur d'accès à ce fichier au cours de l'exécution de cette méthode déclenche une exception `IOException`. Le paramètre `format` est quant à lui destiné à spécifier le format selon lequel les données d'un catalogue doivent être exportées. Son type, `BookCatalogFormat`, correspond à l'interface suivante :

```
public interface BookCatalogFormat
{
  String beginDocument();
  String beginBook();
  String outputTitle(String title);
  String outputAuthor(String firstname, String lastname);
  String outputComment(String comment);
  String endBook();
  String endDocument();
}
```

   Chacune des méthodes de cette interface a pour responsabilité de générer une chaîne de caractères représentant une donnée bien précise d'un catalogue, comme par exemple le titre d'un livre, ou les nom et prénom d'un de ses auteurs, dans le format choisi.

(a) Ecrire le code de la classe `BookCatalog`.

(b) Ecrire le code d'une classe `XMLBookCatalogFormat` implémentant valablement l'interface `BookCatalogFormat`, et représentant un format d'exportation vers un dialecte XML particulier. Une opération d'exportation dans ce format doit produire un fichier respectant les règles suivantes :

- Le fichier commence par `<?xml version="1.0" encoding="UTF-8"?>`, et ouvre ensuite la balise `<library>`.
- Les livres du catalogue sont énumérés les uns à la suite des autres. Les données de chaque livre sont placées entre les balises `<book>` et `</book>`.
- Le titre d'un livre est placé entre les balises `<title>` et `</title>`.
- Les auteurs d'un livre sont énumérés les uns à la suite des autres. Chaque auteur est placé entre les balises `<author>` et `</author>`. Entre ces deux balises, son prénom est placé entre les balises `<first>` et `</first>`, et son nom entre `<last>` et `</last>`.
- Le commentaire associé à un livre est placé entre les balises `<comment>` et `</comment>`.
- Le fichier se termine par la fermeture de la balise `</library>`.

Par exemple, dans ce format, un catalogue contenant uniquement le livre « Data Structures & Algorithms in Java » écrit par Michael T. Goodrich et Roberto Tamassia, avec le commentaire « Un bon livre sur les structures de données », s'exporte de la façon suivante :

```
<?xml version="1.0" encoding="UTF-8" ?>
<library>
  <book>
    <title>Data Structures & Algorithms in Java</title>
    <author><first>Michael T.</first><last>Goodrich</last></author>
    <author><first>Roberto</first><last>Tamassia</last></author>
    <comment>Un bon livre sur les structures de données</comment>
  </book>
</library>
```

Notes :

- On demande que chaque instance d'un catalogue de livres puisse être **clonée en profondeur**.
- Vous êtes libre de développer des classes supplémentaires nécessaires à votre solution.
- Veillez à signaler les situations d'erreur par des exceptions implémentées par vos soins.
- Il **n'est pas nécessaire** de récupérer l'ensemble des exceptions déclenchées lors de l'exécution de votre code, **sauf** les erreurs d'accès à un fichier dans la méthode `export`.
- Il **n'est pas nécessaire** d'insérer vos classes dans un autre groupe de classes (package) que celui par défaut.
- Une documentation de quelques classes de la bibliothèque standard Java pouvant être utiles à ce problème est fournie en annexe.

2. Répondre aux questions suivantes **en justifiant**. En Java :

   (a) Quelle forme donne-t-on aux méthodes destinées à tester l'équivalence de deux objets ? Quelles propriétés doivent-elles satisfaire ?

   (b) Que produit l'exécution de ce fragment de code ?

```
java.util.Vector<Integer> a, b;
a = new java.util.Vector<Integer>(); a.add(1); a.add(2); a.add(3);
b = new java.util.Vector<Integer>(); b.add(1); b.add(2); b.add(3);
System.out.println(a == b);
```

   (c) Soit l'extrait de code suivant :

```
import java.util.Arrays;
public class Triangle3D
{
  private double[] points;

  public Triangle3D(
    double p1_x, double p1_y, double p1_z,
    double p2_x, double p2_y, double p2_z,
    double p3_x, double p3_y, double p3_z
  )
  {
    points = new double[]{ p1_x, p1_y, p1_z, p2_x, p2_y, p2_z, p3_x, p3_y, p3_z };
  }

  public double[] getPoint(int number) throws ParameterOutOfBoundsException
  {
    if ((number < 1) || (number > 3))
      throw new ParameterOutOfBoundsException();

    int offset = (number - 1) * 3;
    return Arrays.copyOfRange(points, offset, offset + 3);
  }
}
```

   Transformer la classe `Triangle3D` de façon à implémenter le test d'équivalence entre deux de ses instances.

   Note : Il n'est pas nécessaire de recopier dans votre réponse ce qui n'a pas été modifié.

3. Un chercheur travaille régulièrement avec des matrices possédant exactement 8 colonnes et un nombre de lignes variable, souvent très grand. Il a développé la classe suivante, permettant d'appliquer un filtre exponentiel séparé à chaque colonne d'une matrice :

```
public class MatrixFilter
{
  protected double[][] data;

  ...

  public double[] expFilter(double alpha) throws ParameterOutOfBoundsException
  {
    if ((alpha < 0.0) || (alpha > 1.0))
      throw new ParameterOutOfBoundsException();

    double[] mean = new double[8];
    for (int col = 0; col < 8; ++col)
    {
      mean[col] = data[0][col];
      for (int row = 1; row < data.length; ++row)
        mean[col] = (1.0 - alpha) * mean[col] + alpha * data[row][col];
    }

    return mean;
  }
}
```

Afin d'améliorer l'efficacité de ce code, le chercheur souhaite exploiter les 8 cœurs du processeur de son ordinateur en effectuant en parallèle les opérations de filtrage de chaque colonne. Autrement dit, 8 contextes d'exécution distincts seraient créés, appliquant chacun le filtre exponentiel à une colonne unique de la matrice.

(a) Ecrire le code d'une sous-classe `ParallelMatrixFilter` de `MatrixFilter`, implémentant l'amélioration discutée ci-dessus tout en présentant la même interface que sa classe parente.

(b) Expliquer l'utilité du mot-clé `synchronized` en Java.

(c) Est-il nécessaire d'utiliser ce mot-clé dans votre solution à la sous-question (a) ? **Justifier**.

Documentation Java

Cette annexe fournit la documentation de quelques classes de la bibliothèque standard Java pouvant être utiles à la résolution de l'exercice 1.

Note : l'utilisation de ces classes et méthodes n'est pas obligatoire !

Classe `java.util.Vector<E>` **:** file de données (politique FIFO)

- `void add(E e)` : ajoute la référence `e` en fin de file.
- `E get(int i)` : retourne la référence stockée à l'indice `i` (le premier élément possédant un indice égale à 0).
- `E set(int i, E e)` : remplace la référence stockée à l'indice `i` par la référence `e` et retourne l'ancienne référence.
- `E remove(int i)` : supprime la référence stockée à l'indice `i` et retourne cette référence. Tous les éléments situés à droite de celui qui est supprimé sont décalés d'une position vers la gauche.
- `boolean isEmpty()` : indique si la file est vide.
- `int size()` : retourne la taille de la file.

Classe `java.io.FileWriter` **:** écriture dans un fichier

- `FileWriter(String f)` : instancie un objet ouvrant en écriture le fichier `f`.
- `void write(String s) throws IOException` : écrit le contenu de `s` dans le fichier `f`.
- `void close() throws IOException` : ferme le fichier `f`.

Manipulation de tableaux

- La variable `length` d'un objet tableau contient la taille de celui-ci.