Examen (avec document)

Préambule : Répondre de manière concise et précise aux questions. Ne pas mettre de commentaires de documentation sauf s'ils sont nécessaires à la compréhension.

Répondre directement dans le sujet quand c'est possible. Sinon, mettre une marque sur le sujet (par exemple le numéro de l'exercice suivi d'une lettre majuscule : 1A, 1B, 2A, etc) et d'indiquer sur la copie la marque avec le texte (ou le code) associé.

Les exercices sont relativement indépendants.

Barème indicatif :

exercice	1	2	3	4
points	3	5	7	5

JMX (Java Management Extensions) est une API pour Java permettant de superviser une application en cours d'exécution. JMX est activé par défaut depuis la version 6 de J2SE. J2SE propose l'application JConsole qui permet la supervision d'une application Java. Bien sûr, c'est l'application qui décrit ce qui peut être supervisé au moyen de ressources appelées MBean (Managed Bean).

Une façon simple de mettre en œuvre les MBean est de s'appuyer sur des conventions de nommage. Ce sont les MBean standard. Plus précisément, pour décrire une ressource R, on définit une classe R qui réalise l'interface RMBean. L'interface décrit ce qui pourra être fait lors de la supervision. On distingue alors les propriétés (aussi appelées attributs) et les opérations. Une propriété peut être en lecture seule ou en lecture et écriture. Une propriété `prop` est en lecture s'il existe dans l'interface une méthode de la forme `T getProp()`. La propriété est aussi en écriture s'il existe une méthode de la forme `void setProp(T nv)`. Les autres méthodes de l'interface RMBean correspondent aux opérations disponibles sur la ressource.

Par exemple, la ressource Memoire est spécifiée par l'interface MemoireMBean (listing 1). Elle définit une propriété `valeur` en lecture et écriture et deux propriétés `nbAcces` et `nbModif` en lecture seule. Trois opérations sont définies : `unset`, `append` et `prepend`.

Ces ressources doivent être enregistrées dans un serveur pour ensuite être retrouvées et utilisées par l'application de supervision. Cet aspect n'est pas traité dans ce sujet.

Listing 1 – L'interface MemoireMBean

```
1  public interface MemoireMBean {
2      String getValeur();
3      void setValeur(String valeur);
4      int getNbModif();
5      int getNbAcces();
6      void unset();
7      void append(String suffix);
8      void prepend(String prefix);
9  }
```

Exercice 1 : Un objet est-il une ressource

Écrire une réalisation de l'interface `IMBeanGetter` (listing 2). Sa méthode retourne l'objet de type `Class` qui représente l'interface de l'objet o. Si l'objet n'est pas un MBean, `null` est retourné.

Listing 2 – L'interface IMBeanGetter

```
1  public interface IMBeanGetter {
2      Class<?> getMBeanInterface(Object o);
3  }
```

Exercice 2 : Éléments d'une ressource

Compléter la classe `MBeanInspecteur` (listing 4) qui réalise l'interface `MBeanContenu` (listing 3). Le constructeur de cette classe prend en paramètre l'objet correspondant à la ressource dont les propriétés et opérations doivent être retrouvées.

Listing 3 – L'interface MBeanContenu

```
1  import java.lang.reflect.Method;
2  import java.util.*;
3
4  public interface MBeanContenu {
5      /** Obtenir la liste de méthodes contient la méthode d'accès
6       * (getXxx) en premier position et, si elle existe, la méthode
7       * de modification (setXxx) en deuxième position.
8       * @return les propriétés d'un MBean.
9       */
10     Map<String, List<Method>> getProprietes();
11
12     /** Obtenir les opérations du MBean.
13      * @return les opérations.
14      */
15     List<Method> getOperations();
16 }
```

Listing 4 – Squelette de la classe MBeanInspecteur

```
1  import java.util.*;
2  import java.lang.reflect.*;
3
4  public class MBeanInspecteur implements MBeanContenu {
5
6      /** La propriété correspondant au nom d'une métode. */
7      private static String getPropertyName(String name) {
8          String fin = name.substring(4);
9          char initiale = Character.toLowerCase(name.charAt(3));
10         return initiale + fin;
11     }
12
13     public MBeanInspecteur(Class<?> mbean) {
14     }
15
16 }
```

FIGURE 1 – L'interface utilisateur avant appui sur *new*

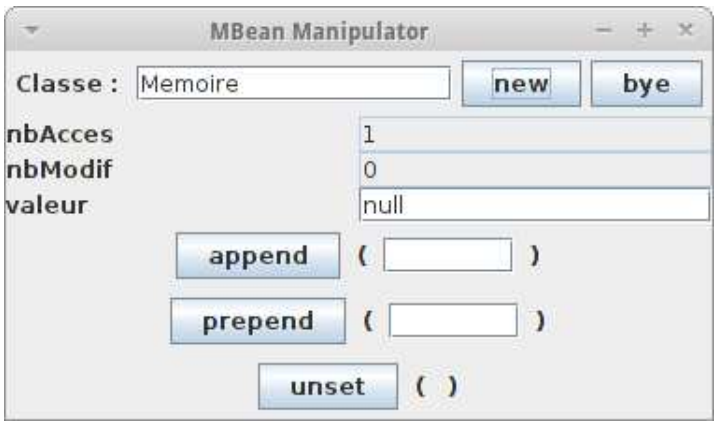

FIGURE 2 – L'interface utilisateur sur un objet Memoire

Exercice 3 : Outil de supervision Swing

L'outil permet de saisir le nom d'une classe (figure 1). Quand l'utilisateur appuie sur le bouton *new*, un objet est créé comme instance de la classe dont le nom a été saisi. On suppose que cet objet correspond à une ressource et est donc un MBean. Un MBean doit avoir un constructeur par défaut. L'application affiche alors les propriétés de ce MBean avec leur valeur ainsi que les opérations disponibles avec une zone de saisie pour chaque paramètre (figure 2).

L'utilisateur peut alors changer la valeur d'une propriété (sauf si elles sont en lecture seule [1]). Il peut aussi renseigner les paramètres d'une méthode (pour simplifier on supposera que ce sont toujours des String) et cliquer sur le nom de la méthode pour exécuter cette méthode.

Quand on change la valeur d'une propriété ou qu'on exécute une méthode, il faut que toutes les valeurs des propriétés soient affichées de nouveau.

3.1 En annotant la figure 2, expliquer comment construire l'interface utilisateur.

3.2 Expliquer comment faire pour mettre à jour la valeur des propriétés quand une opération est exécutée ou la valeur d'une propriété changée.

3.3 Compléter la classe (listing 5). Pour simplifier, on ne rendra actif aucun bouton à l'exception des boutons *new* et *bye*.

Listing 5 – Squelette de la classe MBeanUI

```
1  import javax.swing.*;
2  import java.awt.*;
3  import java.awt.event.*;
4  import java.util.*;
5  import java.lang.reflect.Method;
6
7  public class MBeanUI extends JPanel {
8
9      final private JFrame mainFrame;
10     final private JTextField nomClasse = new JTextField(15);
11     final private JButton newButton = new JButton("new");
12     final private JButton byeButton = new JButton("bye");
13     private Object object;
14
15
16     public MBeanUI(JFrame mainFrame) {
17         this.nomClasse.setText("Memoire");
18         this.mainFrame = mainFrame;
19
20
21     }
22
23
24
25     public static void main(String[] args) {
26         EventQueue.invokeLater(new Runnable() {
27             public void run() {
28                 JFrame fenetre = new JFrame("MBean Manipulator");
29                 Container contenu = fenetre.getContentPane();
30                 fenetre.add(new MBeanUI(fenetre));
31                 fenetre.pack();
32                 fenetre.setVisible(true);
33             }
34         });
35     }
36
37 }
```

1. On peut utiliser setEditable(false) pour désactiver l'édition d'un JTextField.

Exercice 4 : Sérialisation en XML

On souhaite sérialiser un MBean en XML. Compléter la classe `MBeanSerialiseur` (listing 6) pour obtenir comme résultat le fichier XML du listing 7. Donner aussi la DTD `mbean.dtd`.

Listing 6 – Squelette de la classe MBeanSerialiseur

```
1  import java.io.*;
2  import org.jdom.*;
3  import org.jdom.output.*;
4  import java.util.*;
5  import java.lang.reflect.*;
6
7  public class MBeanSerialiseur {
8
9      public void serialiser(Object o, OutputStream out) {
10     }
11
12     private static void ecrire(Document document, OutputStream out) {
13         try {
14             XMLOutputter sortie = new XMLOutputter(Format.getPrettyFormat());
15             sortie.output(document, out);
16         } catch (java.io.IOException e) {
17             throw new RuntimeException("Erreur␣sur␣écriture␣:␣", e);
18         }
19     }
20
21     public static void main(String[] args) throws Exception {
22         Memoire m = new Memoire();
23         m.setValeur("fin");
24         m.getValeur();
25         m.append("i␣!");
26         m.prepend("Presque␣");
27         new MBeanSerialiseur().serialiser(m, new FileOutputStream("memoire.xml"));
28     }
29
30 }
```

Listing 7 – Le fichier XML obtenu

```
1  <?xml version="1.0" encoding="UTF-8"?>
2  <!DOCTYPE mbean SYSTEM "mbean.dtd">
3
4  <mbean class="MemoireMBean">
5    <property name="nbAcces">1</property>
6    <property name="nbModif">3</property>
7    <property name="valeur">Presque fini !</property>
8  </mbean>
```