CORRIGÉ DES QUESTIONS PRÉPARATOIRES À L'EXAMEN

Question 1.
Écrivez un programme en Java qui utilise l'API DOM et modifie un fichier XML de manière à ce que le préfixe « foo » soit ajouté aux noms de tous les éléments.

Le fichier

```
<element><a></a></element>
```

deviendra donc

```
<fooelement><fooa></fooa></fooelement>.
```

Vous pouvez supposer que le document ne contient que des éléments, sans attributs, sans espaces de noms.

Réponse.

```
import org.w3c.dom.*;
import java.io.*;
import javax.xml.parsers.*;
import javax.xml.transform.*;
import javax.xml.transform.dom.*;
import javax.xml.transform.stream.*;

public class Question1 {
      /**
       * Cette méthode parcours récurssivement l'arbre DOM et ajoute
       * le préfixe à chauque élément noeud
       * @param node
       * @param i
       * @param prefix
       */
      public static void traite(Node node, int i, String prefix) {
            if (node.getNodeType()==node.ELEMENT_NODE){
              System.out.println("Nom: "+ node.getNodeName());
              Document doc = node.getOwnerDocument();
              doc.renameNode(node, null, prefix+node.getNodeName());
              System.out.println("Nom après l'jout du prefix : "+
node.getNodeName());
            }
              NodeList nl = node.getChildNodes();
              if(nl != null) {
                    for (int k = 0; k < nl.getLength(); ++k) {
                          traite(nl.item(k),i+2,prefix);
                    }
              }
      }

  public static void main(String[] args) throws Exception {

       if (args.length > 0 ){
            String prefix="foo";
          DocumentBuilderFactory factory =
DocumentBuilderFactory.newInstance();
          DocumentBuilder parser = factory.newDocumentBuilder();
          Document doc = parser.parse(args[0]);
          traite(doc,0,prefix);
          TransformerFactory tfact = TransformerFactory.newInstance();
```

```
            Transformer transformer = tfact.newTransformer();
            transformer.setOutputProperty("encoding", "ISO-8859-1");
            DOMSource source = new DOMSource(doc);
            FileWriter fw = new FileWriter("foo.xml");
            StreamResult result = new StreamResult(fw);
            transformer.transform(source, result);
          }else{
                System.out.println("Usage : java Question1 document.xml");
                System.exit(0);
          }
    }
}
```

Question 2.
Utilisez XSLT pour faire la transformation. (Un indice : pensez à xsl:element.)

Réponse.

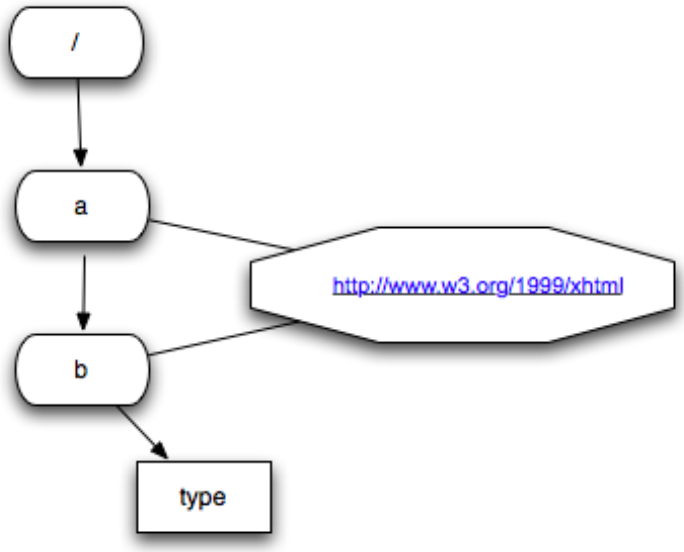
```
<xsl:stylesheet xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
xmlns:redirect="org.apache.xalan.xslt.extensions.Redirect"
extension-element-prefixes="redirect" version="1.0">
<xsl:output method="xml" version="1.0" indent="yes" encoding="ISO-8859-1"/>
<xsl:template match="/">
            <xsl:apply-templates />
</xsl:template>
<xsl:template match="*">
      <xsl:element name="{concat('foo',name(.))}">
            <xsl:apply-templates />
      </xsl:element>
</xsl:template>
</xsl:stylesheet>
```

Question 3.
Dessinez l'arbre DOM du fichier XML suivant.

```
<a  xmlns="http://www.w3.org/1999/xhtml"><b type="x" /></a>
```

Pour chaque élément et attribut, donnez son nom et son espace de noms. (Un indice : attention à l'espace de noms de l'attribut.)

Réponse.



Question 4.
Donnez une DTD équivalente au fichier Relax NG suivant.

```
element p {
   (element c {text}, element d{text}*)?
}
```

Réponse.

```
<!Element p(c, d*) ? >
<!Element c (#PCDATA) >
<!Element  d (#PCDATA) >
```

Question 5.
Écrivez en RDF/XML les triplets suivants.

sujet : http://www.com.com/jeanne
prédicat : http://www.w3.org/1999/02/22-rdf-syntax-ns#type
objet : http://www.com.com/maman

sujet : http://www.com.com/maman
prédicat : http://action.com/etre
objet : http://www.com.com/femme

sujet : http://www.com.com/enfant
prédicat : http://action.com/avoir
objet : http://www.com.com/maman

Réponse.

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
xmlns="http://action.com/" >
<rdf:Description rdf:about="http://www.com.com/jeanne">
   <rdf:type rdf:resource="http://www.com.com/maman" />
</rdf:Description>
<rdf:Description rdf:about="http://www.com.com/maman">
   <etre rdf:resource="http://www.com.com/femme" />
</rdf:Description>
<rdf:Description rdf:about="http://www.com.com/enfant">
   <avoir rdf:resource="http://www.com.com/maman" />
</rdf:Description>
</rdf:RDF>
```

Question 6.
Définissez et expliquez les différences entre XPath, XLink, XQuery et XSLT.

Réponse.

XPath est un langage W3C qui permet d'extraire des données à partir d'un document XML. Il est utilisé pour adresser les éléments dans un document XML à l'aide des expressions de chemins.

Le résultat d'une expression XPath peut être : un ensemble d'éléments ou de nœuds, une chaîne de caractères, un nombre ou un booléen.

XLink est une spécification du W3C (appelé parfois *XLL* pour XLink Language). Il permet de créer des liens entre des fichiers XML ou des fragments de fichiers XML (grâce à XPointer). Contrairement aux liens entre fichiers HTML, XLink permet de créer des liens liant plus de deux fichiers (liens plus sophistiqués).

XQuery est un langage de W3C à la manière de SQL. Il sert à extraire des données à partir de bases de données XML. Il permet entre autres de joindre, d'extraire et d'interroger des documents XML.

XSLT est un langage recommandé par W3C pour transformer des documents XML en d'autres documents XML, HTML, etc.

La différence principale entre ces langages réside dans leur rôle. XSLT est conçu comme une feuille de style qui transforme des documents XML afin qu'ils soient compréhensibles sur un écran, le web, le papier, etc. XQuery, quant à lui, interroge et manipule une collection de documents XML comme SQL. XSLT et XQuery incluent XPath pour accéder aux composants des documents XML.

XPath sert donc comme un support pour XSLT et XQuery. Finalement, XLink ne permet ni d'extraire ou de transformer des données XML, mais plutôt d'insérer des liaisons entre ces données.

Question 7.
Que signifie l'expression Xpath suivante?
//table[@border="0" and @cellspacing="0"] | //table[@border="1" and @cellspacing="1"]

Réponse.
L'union des éléments table dont la valeur de l'attribut border est 0 et la valeur de l'attribut cellspacing est 0 ou la valeur de l'attribut border est 1 et la valeur de l'attribut cellspacing est 1.

Question 8.
Que retourne l'expression XPath suivante : //table[@border]

Réponse.
Cette expression retourne les éléments table (descendants du nœud contexte) qui ont un attribut border.

Question 9.
Que retourne l'expression XPath suivante : //table[@border]/@cellspacing

Réponse.
Cette expression retourne les valeurs de l'attribut cellspacing des éléments table (descendants du nœud contexte) qui ont un attribut border.

Question 10.
Que retourne l'expression XPath suivante : //table[@border]/*/*[@style]

Réponse.
Cette expression retourne les éléments qui ont un attribut style et qui sont à leur tour les petits-fils des éléments table (descendants du nœud contexte) ayant un attribut border.

Question 11.
Que retourne l'expression XPath suivante : //table[@border]/*/*[not(@style)]

Réponse.
Cette expression retourne les éléments qui n'ont pas un attribut style et qui sont, à leur tour, les fils des éléments table (descendants du nœud contexte) ayant un attribut border.

Question 12.
Écrivez un programme Java qui applique l'expression XPath de la question précédente à un fichier chargé à partir du disque.

Réponse.

```
import javax.xml.xpath.*;
import javax.xml.parsers.*;
import org.w3c.dom.*;

public class Question12 {
      public static void main(String[] args) throws Exception {
            if (args.length > 0 ){
            try{
```

```
                    DocumentBuilderFactory                  dbfact                  =
DocumentBuilderFactory.newInstance();
                    DocumentBuilder builder = dbfact.newDocumentBuilder();
                    Document document = builder.parse(args[0]);
                    XPathFactory fact = XPathFactory.newInstance();
                    XPath xpath = fact.newXPath();
                    String                          results                         =
xpath.evaluate("//table[@border]/*/*[not(@style)]", document);
                    System.out.println(results);
              } catch (Exception e){
                    e.printStackTrace();
              }
              } else{
                      System.out.println("Usage : java Question12 document.xml");
                      System.exit(0);
              }
        }

}
```

Exemple de document.xml

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<document>
<table border="0">
      <tr>
            <td style="color:red;">cet element ne doit pas etre retourne 1</td>
            <td>cet element doit etre retourne 1</td>
            <td>cet element doit etre retourne 2 ?</td>
      </tr>
</table>
<table>
      <tr>
            <td style="color:red;">cet element ne doit pas etre retourne 1</td>
            <td>cet element ne doit pas etre retourne 1</td>
            <td>cet element ne doit pas etre retourne 2 ?</td>
      </tr>
</table>

</document>
```

L'exécution de la commande java Question12 document.xml retourne

```
cet element doit etre retourne 1
```

Question 13.

Expliquez ce que fera le fichier XSLT suivant.

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<xsl:stylesheet version="1.0"
xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
xmlns:dc="http://purl.org/dc/elements/1.1/"
>
<xsl:template match="/">
- <xsl:for-each select="@type" >
<li>
<xsl:value-of select="." />
</li><xsl:text>
</xsl:text>
</xsl:for-each>
</xsl:template>
```

```
</xsl:stylesheet>
```

Réponse.

Le fichier XSLT met les valeurs de l'attribut type entre <li> et

Question 14.

Écrivez une expression XPath qui compte le nombre d'éléments ayant été déclarés comme contenant du texte en langue française.

Réponse.

Count(//*[xml:lang='fr']) (je n'ai pas encore vérifier ma réponse)

Question 15.

Nommez au moins trois modèles de programmation XML utilisés en Java.

Réponse.

Voici quatre modèles avec leur notice explicative (Il suffit d'en citer trois).

Traitement du XML comme du texte

Un fichier XML est traité comme un fichier texte. Les *expressions régulières* (voir le paquetage « java.util.regex » en Java) peuvent être utilisées pour parcourir le texte.

Traitement événementiel

Le traitement événementiel est une approche très simple pour traiter un document XML, mais qui peut être difficile pour le programmeur. En fait, une API (par exemple, API SAX) est utilisée pour lire en séquence le document XML. Chaque fois qu'une nouvelle balise ou tout autre élément significatif (instruction XML, contenu XML, etc.) est rencontré, un « événement » est généré.

Traitement avec itérateurs

Au lieu d'attendre passivement de recevoir des événements pour les traiter, le flux XML est traité itérativement dans une simple boucle. La *Streaming API for XML* (StAX) permet de réaliser ce type de traitement.

Traitement avec modèle en arbre

Un document XML peut être vu comme un arbre avec un élément-racine contenant lui-même d'autres éléments qui, à leur tour, contiennent des éléments, et ainsi de suite. Ainsi, en utilisant le traitement événementiel par exemple, une librairie peut d'abord lire le document XML et créer un modèle en arbre. Le modèle en arbre XML le plus connu et qui sert de référence est le *Document Object Model* (DOM).

Question 16.
Définissez AJAX et expliquez pourquoi il s'agit d'une technique importante en développement web. Quel est le lien entre AJAX et DOM?

Réponse.

AJAX, ou Asynchronous JavaScript And XML (« XML et Javascript asynchrones »), est un acronyme désignant une méthode informatique de développement d'applications web. Dans une application web classique, une requête envoyée à un serveur HTTP agit en fonction de l'action et des données reçues et renvoie une nouvelle page. Ce type de fonctionnement recharge systématiquement la totalité de la page même si qu'une partie de cette page est requise. Les applications utilisant les techniques AJAX, quant à elles, peuvent envoyer des requêtes au serveur HTTP pour récupérer uniquement les données nécessaires en utilisant la requête HTTP XMLHttpRequest, et en utilisant la puissance des feuilles de style (CSS) ainsi que le langage Javascript côté client pour interpréter la réponse du serveur HTTP. Les applications sont alors plus réactives.

En AJAX, on utilise des scripts ECMAScript pour charger des informations à partir d'un serveur donné. Les implémentations ECMAScript supportent bien l'API DOM en général. ECMAScript fait donc le lien entre AJAX et l'API DOM.

Question 17.
Selon le fichier CSS suivant, quelle sera la couleur d'un élément p?

```
* {color: black;}
p {color:olive;}
body p {color:blue;}
body > p {color: red; }
body > p {color: green; }
* {color: yellow;}
```

Réponse.

Emplacement de p	Couleur de p	Explication
p contenu immédiatement dans body	green	Les deux règles : body > p {color: red; } et body > p {color: green; } s'appliquent mais c'est la dernière qui prévaut.
p contenu dans body mais pas immédiatement	blue	La règle body p {color:blue;} s'applique.
Autres emplacements	olive	Les trios règles : * {color: black;}, p {color:olive;} et * {color: yellow;} s'appliquent, mais c'est la plus spécifique qui prévaut.

Question 18.
Selon le modèle de boîte CSS du W3C, quelle est la hauteur (*height*) minimale d'un objet ayant les propriétés CSS « padding: 10px; » et « margin: 12px; »?

Réponse.

La hauteur minimale est de zéro.

Question 19.

Donnez un exemple de document XML contenant des attributs ID et des attributs IDREF.

Réponse.

Deux solutions.

1. xml:idref='identificateur' permet de faire référence à une clef.

```
<section xml:id='intro'>
   <titre>introduction à XML</titre>
   ...
</section>
<section>
   <p> Après la section <xref xml:idref='intro'>d'introduction</xref>
   nous allons passer ...
</section>
```

2. Soit le fichier DTD « exemple.dtd » :

```
<!ELEMENT x (a+ , b+, c+,) >
<!ELEMENT a (#PCDATA)>
<!ELEMENT b (#PCDATA)>
<!ELEMENT c (#PCDATA)>
<!ATTLIST a mark ID #REQUIRED>
<!ATTLIST b id ID #REQUIRED>
<!ATTLIST c ref IDREF #REQUIRED>
```

Ci-dessous un exemple d'un fichier XML avec ID et IDREF.

```
<!DOCTYPE x SYSTEM "exemple.dtd">
<x>
  <a mark="a1"/>
  <a mark="a2"/>
  <a mark="a3"/>
  <b id="b001" />
  <c ref="a3" />
</x>
```

Question 20.

Selon le fragment de DTD suivant, qu'est-ce qu'il est permis de mettre dans un élément p? Comment nomme-t-on le concept utilisé ici?

Réponse.

```
<!ENTITY % fontstyle "big | small" >
<!ENTITY % phrase "em | cite" >
<!ENTITY % in "%fontstyle; | %phrase;" >
<!ENTITY % In "(#PCDATA |%in;)*" >
<!ELEMENT p %In;>
```

```
<!ELEMENT p (#PCDATA | big | small | em | cite )*>
```

Question 21.

Expliquez les expressions XPath suivantes et décrivez le résultat selon la théorie des ensembles (union, intersection et complément).

```
//*[@s="1"]|//*[@t="1"]
//*[@s="1"][count(.|//*[@t="1"])=count(//*[@t="1"])]
//*[@s="1"][count(.|//*[@t="1"])!=count(//*[@t="1"])]
```

Réponse.

Expression	Valeur retournée
`//*[@s="1"]|//*[@t="1"]`	L'union des éléments dont la valeur des attributs s et t est égale à 1.
`//*[@s="1"][count(.|//*[@t="1"])=count(//*[@t="1"])]`	L'intersection des éléments dont la valeur de s et de t est égale à 1.
`//*[@s="1"][count(.|//*[@t="1"])!=count(//*[@t="1"])]`	Le complément des éléments dont la valeur de s est égale à 1 et la valeur de t est différente de 1.

Question 22.

Comment passe-t-on des paramètres à une feuille de style XSLT à partir d'un document XML?

Réponse.

On peut passer un paramètre à une feuille de style XSLT à l'aide de « xslt-param ». Ci-dessous, un exemple de fichier XML (tiré du cours) dans lequel on passe deux paramètres.

```
<?xml version="1.0" encoding="ISO-8859-1" ?>
  <?xslt-param name="couleur" value="blue"?>
  <?xslt-param name="taille" select="2"?>
  <?xml-stylesheet href="xslt.xml"  type="application/xml"?>
  <facture>
    <montant>10.10</montant>
    <personne>Jean Rochond</personne>
    <raison>Achat d'ordinateur</raison>
</facture>
```

(Optionnel)

La récupération des paramètres se fait à l'aide de « xsl:param ». Ci-dessous, un exemple d'un fichier XSLT récupérant ces deux paramètres (l'attribut select permet d'associer des valeurs par défaut).

```
<?xml version="1.0" encoding="ISO-8859-1"?>
 <xsl:stylesheet version="1.0"
  xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
  <xsl:param name="couleur" select="red">
  <xsl:param name="taille" select="1">
 <xsl:template match="facture">
 <html>
 <head>
 <title>Facture de <xsl:value-of select="personne" /></title>
 </head>
 <body>
 <p style="color:{$couleur}; font-size:{$taille}em">Ceci est
 une facture pour <xsl:value-of select="personne" />
  de <xsl:value-of select="montant" />$ pour:
 <xsl:value-of select="raison" />.</p>
 </body>
 </html>
 </xsl:template>
 </xsl:stylesheet>
```

Question 23.

Écrivez un programme XQuery qui donne la liste des éléments distincts utilisés dans un document XML « etu.xml ».

```
distinct-values(for $s in doc("etu.xml")//*
  return name($s)
)
```

Question 24.

Quels sont les langages Turing-complet dans cette liste : Java, ECMAScript, CSS, XSLT, XQuery, XPath ?

Java, ECMAScript, XSLT et XQuery sont Turing-complet. CSS et XPath ne le sont pas.