Procesamiento de Formatos en Aplicaciones Telemáticas - Examen convocatoria ordinaria, curso 2014/15
Grado en Ingeniería Telemática
Depto. de Ingeniería Telemática
Universidad Carlos III de Madrid

Duración: 1 hora 15 minutos
Puntuación: 30 puntos
Nota: Se pueden usar libros y apuntes

EJERCICIO 1 (18 puntos)

Utilizando el siguiente DTD como dato:

```
<!ELEMENT personal (empleado+)>
<!ELEMENT empleado (nombre,telefono,correo)>
<!ATTLIST empleado id ID #REQUIRED
                   url CDATA #IMPLIED
                   cargo (socio|gerente|consultor) "consultor">
<!ELEMENT nombre (#PCDATA)>
<!ELEMENT telefono (#PCDATA)>
<!ELEMENT correo (#PCDATA)>
```

Se pide implementar un método en Java cuyo prototipo es:

```
String ordinaria(Document doc, String referencia)
```

que dado un objeto `Document doc` que representa un documento XML válido de acuerdo con el DTD definido anteriormente devuelva el contenido del elemento `nombre` que sea hijo de un elemeto `empleado` cuyo atributo `id` tome el valor que se pasa en el argumento `referencia`. El método devolverá "" si no se encuentra el elemento `nombre` pedido.

Por ejemplo, si `doc` representase el siguiente documento XML:

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE personal SYSTEM "ejercicio.dtd">
<personal>
  <empleado id="c1">
    <nombre>Juan Pérez</nombre>
    <telefono>600123456</telefono>
    <correo>juan@pfat.es</correo>
  </empleado>
  <empleado id="c2">
    <nombre>Luis Muñoz</nombre>
    <telefono>609456123</telefono>
    <correo>luis@pfat.es</correo>
  </empleado>
</personal>
```

`ordinaria(doc, "c2")` devolvería "`Luis Muñoz`".

Se puede suponer que los elementos `nombre` contienen un único nodo hijo de tipo `Text`.

Solución:

```
    private static String ordinaria(Document doc, String referencia) {
        Element el1, el2, docEl;
        NodeList nL, nL2;
        int i, j, s1, s2;
        Node n1, n2;
```

```
        docEl= doc.getDocumentElement();

        nL= docEl.getChildNodes();
        s1= nL.getLength();

        if (s1>0) {
            for (i=0; i<s1; i++) {
                n1= nL.item(i);
                if (n1.getNodeType() == Node.ELEMENT_NODE) {
                    el1= (Element) n1;
                    if (el1.getAttribute("id").equals(referencia)) {
                        nL2= el1.getChildNodes();
                        s2=nL2.getLength();
                        for(j=0; j<s2; j++) {
                            n2= nL2.item(j);
                            if (n2.getNodeType() == Node.ELEMENT_NODE) {
                                el2= (Element) n2;
                                if (el2.getNodeName().equals("nombre")) {
                                    System.out.println("Encontrado");
                                    return el2.getFirstChild().getNodeValue();
                                }
                            }
                        }
                    }
                }
            }
        }
        return "";
    }
```

EJERCICIO 2 (12 puntos)

Utilizando el siguiente DTD para representar matrices como dato:

```
<!ELEMENT matrixE (filaE+)>

<!ELEMENT filaE (celdaE+)>

<!ELEMENT celdaE (#PCDATA)>
```

Escriba una hoja de estilo XSLT que dado un documento XML válido de acuerdo con este DTD devuelva un documento XML válido para el mismo DTD que devuelva la misma matriz que representaba el documento original salvo por lo siguiente:

- Las celdas cuyo valor en la matriz original era 0 pasan a contener un 1.
- Las celdas cuyo valor en la matriz original era 1 pasan a contener un 0.
- El valor de las celdas restantes debe ser el mismo que en la matriz original.

Por ejemplo, si la matriz original fuese:

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE matrixE SYSTEM "matrix.dtd">

<matrixE>
<filaE><celdaE>1</celdaE><celdaE>0</celdaE><celdaE>2</celdaE></filaE>
<filaE><celdaE>3</celdaE><celdaE>1</celdaE><celdaE>5</celdaE></filaE>
<filaE><celdaE>0</celdaE><celdaE>1</celdaE><celdaE>2</celdaE></filaE>
</matrixE>
```

El resultado de aplicar la hoja de estilo debería ser:

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE matrixE SYSTEM "matrix.dtd">

<matrixE>
<filaE><celdaE>0</celdaE><celdaE>1</celdaE><celdaE>2</celdaE></filaE>
<filaE><celdaE>3</celdaE><celdaE>0</celdaE><celdaE>5</celdaE></filaE>
```

```
<filaE><celdaE>1</celdaE><celdaE>0</celdaE><celdaE>2</celdaE></filaE>
</matrixE>
```

No se preocupe por como queda indentado el documento XML resultante.

Solución:

```
<?xml version="1.0"?>

<xsl:stylesheet xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
  version="1.0">

<xsl:output method="xml" encoding="ISO-8859-1"
  doctype-system="matrix.dtd" indent="yes"/>

<xsl:strip-space elements="*" />

<xsl:template match="*">
<xsl:copy>
<xsl:apply-templates/>
</xsl:copy>
</xsl:template>

<xsl:template match="celdaE">
<celdaE>
<xsl:choose>
<xsl:when test=".=0">1</xsl:when>
<xsl:when test=".=1">0</xsl:when>
<xsl:otherwise>
  <xsl:value-of select="."/>
</xsl:otherwise>
</xsl:choose>
</celdaE>
</xsl:template>
</xsl:stylesheet>
```

Solución mejorada

```
public static String ordinaria(Document doc, String referencia) {
    try {
        return doc.getElementById(referencia).getElementsByTagName("nombre").item(0).getFirstChild().getNodeValue();
    } catch (Exception e) {
        return "";
    }
}
```