Procesamiento de Formatos en Aplicaciones Telemáticas - Examen convocatoria extraordinaria, curso 2015/16
Grado en Ingeniería Telemática
Depto. de Ingeniería Telemática
Universidad Carlos III de Madrid

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

En los dos ejercicios se utilizará el siguiente DTD:

```
<!ELEMENT matrixE (filaE+)>
<!ATTLIST matrixE
totalCeldas CDATA #IMPLIED>


<!ELEMENT filaE (celdaE+)>
<!ATTLIST filaE
celdas CDATA #IMPLIED>

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

Un ejemplo de documento XML válido de acuerdo con este DTD sería el siguiente:

```
<?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>
```

EJERCICIO 1 (15 puntos)

Se pide implementar un método en Java cuyo prototipo es:

```
int extraordinaria(Document doc)
```

que dado un objeto `Document doc` que representa un documento XML válido de acuerdo con el DTD definido anteriormente devuelva la suma de los valores de las celdas del primer hijo `filaE` del elemento raiz del documento XML dado.

Por ejemplo, suponiendo que `doc` representase el documento XML dado al inicio del enunciado, `extraordinaria(doc)` devolvería `3`.

Se puede suponer que los elementos `celdaE` contienen un único nodo hijo de tipo `Text` y que la cadena de caracteres asociada a dicho nodo representa un número entero.

Puede ser de utilidad el siguiente método de la clase Java `Integer`:

```
static int parseInt(String s)
```

Este método, dado un objeto Java de la clase `String` que representa un número entero, devuelve dicho número entero.

Solución:

```
    int extraordinaria(Document doc) {
        Element docEl;
        NodeList nl1, nl2, nl3;
        int i, j, k, len1, len2, len3;
        Node n1, n2, n3;
        int res= 0;
        docEl= doc.getDocumentElement();
        nl1= docEl.getChildNodes();
        len1=nl1.getLength();
        for (i=0; i<len1; i++) {
            n1= nl1.item(i);
            if (n1.getNodeType()==Node.ELEMENT_NODE) {
                nl2=n1.getChildNodes();
                len2=nl2.getLength();
                for (j=0; j<len2; j++) {
                    n2=nl2.item(j);
                    if (n2.getNodeType()==Node.ELEMENT_NODE) {
                        nl3=n2.getChildNodes();
                        len3=nl3.getLength();
                        for (k=0; k<len3; k++) {
                            n3=nl3.item(k);
                            if (n3.getNodeType()==Node.TEXT_NODE ||
                                n3.getNodeType()==Node.CDATA_SECTION_NODE)
                                res=res + Integer.parseInt(n3.getNodeValue());
                        }
                    }
                }
                return res;
            }
        }
        return -1;
    }
```

EJERCICIO 2 (15 puntos)

Escriba una hoja de estilo XSLT que dado un documento XML válido de acuerdo con el DTD dado al inicio del enunciado devuelva un documento XML igual al documento original salvo por lo siguiente: se suprime la primera fila (y todo su contenido) y la primera celda (y su contenido) de cada una de las filas restantes.

Por ejemplo, para el documento XML proporcionado al principio del enunciado, el resultado de ejecutar la hoja de estilo pedida sería:

```
<?xml version="1.0" encoding="UTF-8"?>

<matrixE>
<filaE><celdaE>1</celdaE><celdaE>5</celdaE></filaE>
<filaE><celdaE>1</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-of select="." />
</xsl:template>

<xsl:template match="matrixE">
<matrixE>
<xsl:apply-templates select="filaE[position()>1]"/>
</matrixE>
</xsl:template>


<xsl:template match="filaE">
<filaE>
  <xsl:apply-templates
    select="celdaE[position()>1]"/>
</filaE>
</xsl:template>

</xsl:stylesheet>
```

Solución mejorada

```
public static int extraordinaria(Document doc) {
    try {
        int res = 0;
        NodeList nl1= ((Element) doc.getElementsByTagName("filaE").item(0)).getElementsByTagName("celdaE");
        for (int i=0; i<nl1.getLength(); i++) {
            res += Integer.parseInt(nl1.item(i).getFirstChild().getNodeValue());
        }
        return res;
    } catch (Exception e) {
        return -1;
    }
}
```