ANSWER KEY UT1

6a. Build a DTD for XML files that store data about a music library.

* A music library consists of any number of songs under albums.
* An album must have a title, artist, year, and two or more tracks. An album may have one or more producers.
* Represent each producer. The “artist” for an album has name and a band name.
* Each track on an album must have a song and a length.
* A song must have a title and may have a year. It may also have information about who wrote the lyrics and one or more composers.

<!ELEMENT musiclibrary(album +)

<!ELEMENT album (artist, track(track+), (producer+))>

<!ATTLIST album

title CDATA #REQUIRED

year CDATA #REQUIRED >

<!ELEMENT artist EMPTY>

<!ATTLIST artist

name CDATA #REQUIRED

bandname CDATA #REQUIRED >

<!ELEMENT track (song, length) >

<!ELEMENT song (composer+) >

<!ELEMENT composer (#PCDATA) >

<!ATTLIST song

title CDATA #REQUIRED

year CDATA #IMPLIED

writer CDATA #IMPLIED >

<!ELEMENT length (#PCDATA) >

6b.

<albums>

<album title=”xxx” year=”2012” >

<artist aname=”aaa” bandname=”bbb” />

<track>

<song title=”hhh” year=”2012” writer=”jjj” />

<composer> composername1 </composer>

<composer> composername2 </composer>

<length>12</length>

</track>

<track>

<song title=”ggg” year=”2012” />

<composer> composername3</composer>

<length>12</length>

</track>

<producer> producername1 </producer>

<producer> producername2 </producer>

</album>

....

</albums>

(or)

<!ELEMENT musiclibrary(album +)

<!ELEMENT album (artist, title, year, track(track+), (producer+))>

<!ELEMENT title (#PCDATA) >

<!ELEMENT artist (name, bandname) >

<!ELEMENT name (#PCDATA) >

<!ELEMENT bandname (#PCDATA) >

<!ELEMENT year (#PCDATA) >

<!ELEMENT track (song, length) >

<!ELEMENT song (stitle, syear, lyricist?, composer+?) >

<!ELEMENT stitle (#PCDATA) >

<!ELEMENT syear (#PCDATA) >

<!ELEMENT lyricist (#PCDATA) >

<!ELEMENT composer (#PCDATA) >

<!ELEMENT producer (#PCDATA) >

<musiclibrary>

<album>

<title>xxxxx</title>

<artist>

<name>yyy</name>

<bandname>zzz</bandname>

</artist>

<year>2012</year>

<track>

<song>aaaa</song>

<stitle>bbbb<\title>

<syear>2012</syear>

<lyricist>cccc</lyricist>

<composer>dddd</lyricist>

<composer>eeee</lyricist>

</song>

<length>05:00</length>

</track>

<track>

<song>

<stitle>

<syear>

<lyricist>

<composer>

<composer>

</song>

<length>

</track>

<producer>

<producer>

</album>

</musiclibrary>

title CDATA #REQUIRED

year CDATA #REQUIRED >

<!ELEMENT artist EMPTY>

<!ATTLIST artist

name CDATA #REQUIRED

bandname CDATA #REQUIRED >

<!ELEMENT track (song, length) >

<!ELEMENT song (composer+) >

<!ELEMENT composer (#PCDATA) >

<!ATTLIST song

title CDATA #REQUIRED

year CDATA #IMPLIED

writer CDATA #IMPLIED >

<!ELEMENT length (#PCDATA) >

7a.

<xsd:element name=”albums” type=”albumsType”/>

<xsd:complexType name=”albumsType”>

<xsd:elemen t name=”album” type=”albumType” minOccurs=”1” maxOccurs=”unbounded”/>

<xsd:complexType name=”albumType”>

<xsd:all>

<xsd:element name=”artist” type=”artistType” minOccurs=”1” maxOccurs=”1”/>

<xsd:element name=”track” type=”trackType” minOccurs=”2” maxOccurs=”unbounded”/>

<xsd:element name=”producer” minOccurs=”1” maxOccurs=”unbounded”/>

<xsd:simpleType>

<xsd:restriction base=”xsd:string”/>

</xsd:simpleType>

</xsd:element>

</xsd:all>

</xsd:complexType>

<xsd:attribute name=”title”>

<xsd:simpleType>

<xsd:restriction base=”xsd:string”/>

</xsd:simpleType>

</xsd:attribute>

<xsd:attribute name=”year”>

<xsd:simpleType>

<xsd:restriction base=”xsd:decimal”/>

</xsd:simpleType>

</xsd:attribute>

<xsd:complexType name=”artistType”>

<xsd:attribute name=”name”>

<xsd:simpleType>

<xsd:restriction base=”xsd:string”/>

</xsd:simpleType>

</xsd:attribute>

<xsd:attribute name=”bandname”>

<xsd:simpleType>

<xsd:restriction base=”xsd:string”/>

</xsd:simpleType>

</xsd:attribute>

</xsd:complexType>

<xsd:complexType name=”trackType”>

<xsd:sequence>

<xsd:element name=”song” type=”songType” minOccurs=”1” maxOccurs=”1”>

<xsd:element name=”length” >

<xs:simpleType>  
    <xs:restriction base="xs:integer">  
      <xs:minInclusive value="5"/>  
      <xs:maxInclusive value="20"/>  
    </xs:restriction>  
  </xs:simpleType>  
</xs:element>

</xsd:sequence>

<xsd:complexType name=”songType”>

<xsd:element name=”composer” minOccurs=”1” maxOccurs=”unbounded”>

<xsd:simpleType>

<xsd:restriction base=”xsd:string”/>

</xsd:simpleType>

</xsd:element>

<xsd:attribute name=”title” >

  <xs:simpleType>  
    <xs:restriction base="xs:string">  
      <xs:pattern value="[a-z][A-Z]"/>  
    </xs:restriction>  
  </xs:simpleType>  
</xs:attribute>

<xsd:attribute name=”year” type=”xsd:positiveInteger”/>

<xsd:attribute name=”writer” type=”xsd:string”/>

</xsd:complexType>

7b. The XML file for 6a is applicable for 7b also.

8b.

1. To find the title of non-fiction books and price more than 50

Solution:

/bookstore/book [@cat != ‘fiction’] / title | /bookstore/book [price > 50]

| - allows to select several location paths

2. To find the titles of textbooks in xml

/bookstore/book [@cat = ‘textbook’]/title

9a. Write java snippet using DOM parser to create the following XML Document

<library>

<book>

<title> xxx </title>

<author> yyy</author>

<publisher> zzz </publisher>

<year> 1998 </year>

</book>

</library>

try {

DocumentBuilderFactory dbFactory =

DocumentBuilderFactory.newInstance();

DocumentBuilder dBuilder =

dbFactory.newDocumentBuilder();

Document doc = dBuilder.newDocument();

// root element

Element rootElement = doc.createElement("library");

doc.appendChild(rootElement);

Element book = doc.createElement("book");

rootElement.appendChild(book);

Element title = doc.createElement("title");

book.appendChild(title);

Element author = doc.createElement("author");

book.appendChild(author);

Element book = doc.createElement("publisher");

book.appendChild(publisher);

TransformerFactory transformerFactory =

TransformerFactory.newInstance();

Transformer transformer =

transformerFactory.newTransformer();

DOMSource source = new DOMSource(doc);

//for storing in a file

StreamResult result =

new StreamResult(new File("C:\\cars.xml"));

transformer.transform(source, result);

// Output to console for testing

StreamResult consoleResult =

new StreamResult(System.out);

transformer.transform(source, consoleResult);

} catch (Exception e) {

e.printStackTrace();

}

9b. Write a java snippet using SAX parser that uses SAX API and handlers to count the no. of items present and print the name of the items.

<Items>

<item id=xx>

<name>Ram</name>

<price> 110</price>

<quantity> 10</quantity>

</item>

<item id=yy>

<name>John</name>

<price> 98</price>

<quantity> 5</quantity>

</item>

<item id=zz>

<name>Prince</name>

<price> 55</price>

<quantity> 6</quantity>

</item>

</items>

Solution:

public class MyHandler extends DefaultHandler

{

Int count=0;

Boolean bname = false;

public void startElement(String uri, String localName,String qName,

Attributes attributes) throws SAXException {

if (qName.equalsIgnoreCase("item")) {

count++;

}

if (qName.equalsIgnoreCase("name")) {

bname = true;

}

}

public void characters(char ch[], int start, int length) throws SAXException {

if (bname) {

System.out.println("Name: " + new String(ch, start, length));

bname = false;

}

}

Public void endDocument( ) throws SAXException

{

System.out.println("No. of items " + count);

}

public class Demo {

public static void main(String argv[]) {

try {

File input = new File(“test.xml”);

SAXParserFactory factory = SAXParserFactory.newInstance();

SAXParser saxParser = factory.newSAXParser();

MyHandler myhandler = new MyHandler();

Saxparser.parse(input, myhandler);

}catch (Exception e) { ... }

a. Consider the following XML document

<productListing title="ABC Products">

 <product>

   <name>Product One</name>

   <description>Product One is an exciting new widget that will

     simplify your life.</description>

   <cost>$19.95</cost>

   <shipping>$2.95</shipping>

 </product>

 <product>

   <name>Product Two</name>

  </product>

 <product>

   <name>Product Three</name>

   <description>This is such a terrific widget that you will

     most certainly want to buy one for your home and another one

     for your office!</p>

   <cost>$24.95</cost>

   <shipping>$4.00</shipping>

 </product>

</productListing>

Write .xsl program with XSLT rules that transform the above XML file to produce HTML document that generates the following output when rendered in a browser.

**Products Details**

Name: Product One

Description: Product One is an exciting new widget that will simplify your life

Cost: $19.95

Shipping: $2.95

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Name: Product Two

Description: This is such a terrific widget that you will most certainly want to buy one for

your home and another one for your office!

Cost: $24.95

Shipping: $4.00

**Solution:**

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

<xsl:template match=”/”>

<html>

<body>

<h2> Product Details </h2>

<xsl:apply-templates />

</body>

</html>

</xsl:template>

<xsl:template match=”product”>

<xsl:apply-templates select=”name” />

<xsl:apply-templates select=”description” />

<xsl:apply-templates select=”cost” />

<xsl:apply-templates select=”shipping” />

</xsl:template>

<xsl:template match=”name”>

Name: <xsl:value-of select”.” />

<br>

</xsl:template>

<xsl:template match=”description”>

Name: <xsl:value-of select”.” />

<br>

</xsl:template>

<xsl:template match=”cost”>

Name: <xsl:value-of select”.” />

<br>

</xsl:template>

<xsl:template match=”shipping”>

Name: <xsl:value-of select”.” />

<br>

</xsl:template>

</xsl:stylesheet>

(or)

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

<html>

<body>

<h2> Product Details </h2>

<xsl:template match=”/”>

<xsl:for-each select =”/productlisting/product”>

Name: <xsl:value-of select =”name” />

<br>

Description: <xsl:value-of select =”description” />

<br>

Cost: <xsl:value-of select =”cost” />

<br>

Shipping: <xsl:value-of select =”shipping” />

<br>

<h2>\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*</h2>

</xsl:for-each>

</body>

</html>

</xsl:template>