第１２回K225（知識プログラミング方法論）演習

2012年11月22日

ライフスタイルデザイン研究センター　金井秀明

1. 次のXML文の要素「book」の属性を要素「book」の子要素に置き換えたXML文書を作成せよ．

```
<book ISBN=”100”>やさしいXML</book>
```

2. 次の(a)から(e)から整形式XML文書を選択せよ．　ただし，正解が複数ある場合には，それらすべてを挙げよ．

```
(a)
<book>
     <code>ID01</code>
</book>
<book>
     <price>2500</price>
</book>
```

```
(b)
<book>
     <code>ID01</code>
     <price>2500</price>
</book>
```

```
(c)
<book/>
     <code>ID01</code>
     <price>2500</price>
</book>
```

```
(d)
<book>
     <code>ID01
     <price>2500
     </code></price>
</book>
```



```
(e)
<book>
     <code>ID01
     <price>2500</price>
     </code>
</book>
```

3. 要素型宣言について正しい説明文を挙げよ．ただし，正解が複数ある場合には，それらすべてを挙げよ．

(1) <!ELEMENT XYZ (a1, b1)>
XYZ要素は，a1要素を子要素に持ち，かつa1要素はb1要素を子要素に持つ．

(2) <!ELEMENT XYZ (a1 | b1)>
XYZ要素は，a1要素かb1要素のどちらか一方を子要素に持つ．

(3) <!ELEMENT XYZ (XYZ)>
XYZ要素は，XYZ要素を子要素に持つ．

(4) <!ELEMENT XYZ (a1?)>
XYZ要素は，a1要素を子要素に持ち，出現回数は０回もしくは１回である．

4. 次のXML文書を表すDTDを作成せよ．

```
<授業>
	<コード>文字データ</コード>
	<名前>文字データ</名前>
	<先生>
		<姓>文字データ</姓>
		<名>文字データ</名>
	</先生>
</授業>
```


5. 次のXML文書を表すDTDを作成せよ．

```
<?xml version=”1.0”, encoding=”Shift_JIS” ?>
<PC>
	<商品名>文字データ</商品名>
	<仕様>
		説明文１
		<CPU>文字データ</CPU>
		説明文２
		<MEMORY>文字データ</MEMORY>
		説明文３
	</仕様>
</PC>
```

ただし，「商品名」は１回出現する．「仕様」は一回出現する．

6. 次のXML文書Aを検証プロセッサ（Validator）で検証した結果について，（1）～（4）から正しい記述を選べ．ただし，正解が複数ある場合には，それらすべてを挙げよ．

XML文書A

```
<!ELEMENT root (flower)>
<!ELEMENT flower (#PCDate)>
<!ENTITY flower SYSTEM ”flower.xml”>
<root>&flower;</root>
```

flower.xml

```
<flower>バラ</flower>
```

(1) 要素名と同じ名前の実体宣言は記述できないため，DTDの文法エラーとなる．

(2) root要素の要素型宣言で指定しているflowerがXMLパーサで要素か実体か判断できないため，エラーとなる．

(3) root要素の内容は，flower要素を記述しなければならないため，「&flower:」の記述は文法エラーとなる．

(4) 妥当である．

7. 次のXML文として，正しくないものをすべて挙げよ．さらにその理由を書け．

(1) <kamoku＞K225＜/kamoku

(2) <_location>電算室</_location>

(3) <jaist.ac.jp>url</jaist.ac.jp>

(4) < kamoku_bangou>K225</ kamoku_bangou>

8. 次のXMLインスタンスのDTDを作成せよ．（配点６点）

```
<M1>
    <学生>
      <番号>文字データ</番号>
      <資格>文字データ</資格>
    </学生>
</M1>
```

ただし，「学生」は0回以上出現する．「番号」は1回出現する．
「資格」は０回以上出現する．

9. 次のXML文書（Exam9.xml）をDOMで読み込み処理を行う．「チーム」要素を扱っているときに，テキスト「インテル」を取得するにはどのメソッドを利用すればよいか．

```
<?xml version=”1.0” encoding=”Shift_JIS” ?>
<長友><チーム>インテル</チーム></長友>
```

10. 次のXML文書（Exam10.xml）がある．文書全体を扱っているときに，DOMのgetElementsByTagName(”a”)メソッドから返されるNodeListに含まれるNode数はいくつか？

```
<?xml version=”1.0” encoding=”Shift_JIS” ?>
<a><a>abc</a><b><a/></b><a></a></a>
```

11. XML文書（Exam11.xml）をSAXで処理を行う際に発生するイベントの順序を書け．ただし，最初に発生するイベントは「startDocument」である．

```
<?xml version=”1.0” encoding=”Shift_JIS” ?>
<長友><チーム>インテル</チーム></長友>
```


12. 次のPractice.xmlから，DOMを使って書籍の平均価格を表示するプログラム（Exam12.java）を完成せよ． ここで，標準出力結果は表示結果11になるようにすること．

Practice.xml

```
<?xml version="1.0" encoding="Shift_JIS" ?>
<books>
  <book>
    <title>入門SQL</title>
    <price>2400</price>
  </book>
  <book>
    <title>やさしいC++</title>
    <price>2500</price>
  </book>
  <book>
    <title>やさしいJava</title>
    <price>2600</price>
  </book>
</books>
```

表示結果12

```
本の冊数＝3
平均の値段=2500
```

Exam12.java

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

class Exam12 {
   public static void main(String args[]) throws Exception {
       int total_of_price = 0;
       int bookcounter = 0;

      //DOMの準備をする
      DocumentBuilderFactory dbf
         = DocumentBuilderFactory.newInstance();
      DocumentBuilder db
         = dbf.newDocumentBuilder();

      //文書を読み込む
      Document doc
         = db.parse(new FileInputStream("Practice.xml"));

      //ルート要素を得る
      Element root = doc.getDocumentElement();

      [                                    ]

}
```


13. 「12」のPractice.xmlを，DOMを使って以下のDTDで定義されるXML文書に変換するプログラム（Exam13.java)を完成せよ．変換結果は標準出力に出力する．

```
<!DOCTYPE titlelist[
 <!ELEMENT titlelist (title+)>
 <!ELEMENT title (#PCDATA)>
]>
```

Exam13.java

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

class Exam13 {
    public static void main(String args[]) throws Exception {
      //DOMの準備をする
      DocumentBuilderFactory dbf
          = DocumentBuilderFactory.newInstance();
      DocumentBuilder db = dbf.newDocumentBuilder();
      //文書を読み込む
      Document doc
            = db.parse(new FileInputStream("Practice.xml"));
      //ルート要素を得る
      Element root = doc.getDocumentElement();

      //XML宣言を作成する
      System.out.println("<?xml version=¥"1.0¥"
                         encoding=¥"Shift_JIS¥" ?>");

      [                                    ]

}
```

14. 次のPractice.xmlから，SAXを使って書籍の平均価格を表示するプログラム（Exam14.java）のExam14Handlerを完成せよ．ここで，標準出力結果は表示結果14になるようにすること．

Practice.xml

```
<?xml version="1.0" encoding="Shift_JIS" ?>
<books>
  <book>
    <title>入門SQL</title>
    <price>2400</price>
  </book>
  <book>
    <title>やさしいC++</title>
    <price>2500</price>
  </book>
  <book>
    <title>やさしいJava</title>
    <price>2600</price>
  </book>
</books>
```

表示結果14

```
XML文書が開始しました.
XML文書が終了しました.
本の冊数＝3
平均の値段=2500
```


Exam14.java

```
import java.io.*;
import javax.xml.parsers.*;
import org.xml.sax.*;
import org.xml.sax.helpers.*;

class Exam14 {
 public static void main(String args[]) throws Exception {
 //SAXの準備をする
  SAXParserFactory spf = SAXParserFactory.newInstance();
  SAXParser sp = spf.newSAXParser();

 //ハンドラを作成する
  Exam14Handler sh = new Exam14Handler();

 //文書を読み込む
  sp.parse(new FileInputStream("Practice.xml"), sh);
 }
}

//ハンドラクラス
class Exam14Handler extends DefaultHandler {
    
}
```

15. 「13」のPractice.xmlを，SAXを使って以下のDTDで定義されるXML文書に変換するプログラム（Exam15.java)を完成せよ．変換結果は標準出力に出力する．

```
<!DOCTYPE titlelist[
 <!ELEMENT titlelist (title+)>
 <!ELEMENT title (#PCDATA)>
]>
```

Exam15.java

```
import java.io.*;
import javax.xml.parsers.*;
import org.xml.sax.*;
import org.xml.sax.helpers.*;

class Exam15 {
 public static void main(String args[]) throws Exception {
 //SAXの準備をする
  SAXParserFactory spf = SAXParserFactory.newInstance();
  SAXParser sp = spf.newSAXParser();

 //ハンドラを作成する
  Exam15Handler sh = new Exam15Handler();

 //文書を読み込む
  sp.parse(new FileInputStream("Practice.xml"), sh);
 }
}

//ハンドラクラス
class Exam15Handler extends DefaultHandler {

    [                                    ]

}
```