分散システム

第四回レポート課題

課題 1

【課題内容】

講義資料（その 2）ページ 18 およびページ 20 に記載されているソースコードを参考に、第二回レポート課題で作成した XML 文書に対応する DOM 木を構築して出力するプログラムを作成せよ。ただし、

- 作成したソースコード
- 実行結果（スクリーンダンプにより）
- 生成された XML ファイルのブラウザによる表示結果（スクリーンダンプにより）

を示すこと。

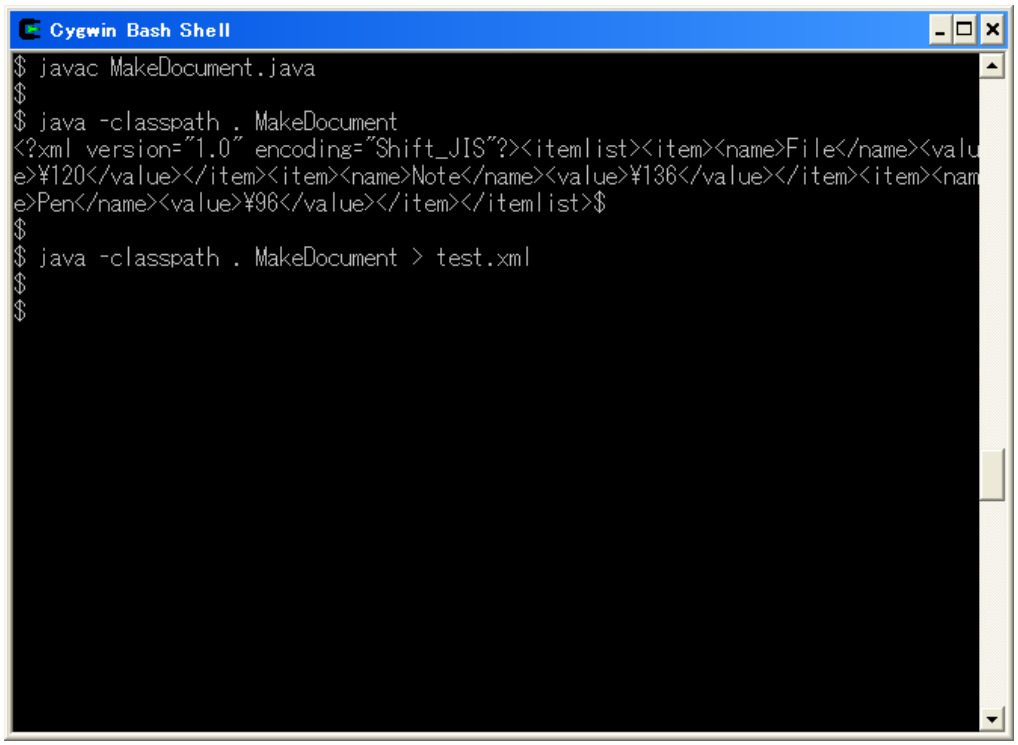
【作成したソースコード】

≪*MakeDocument.java*≫

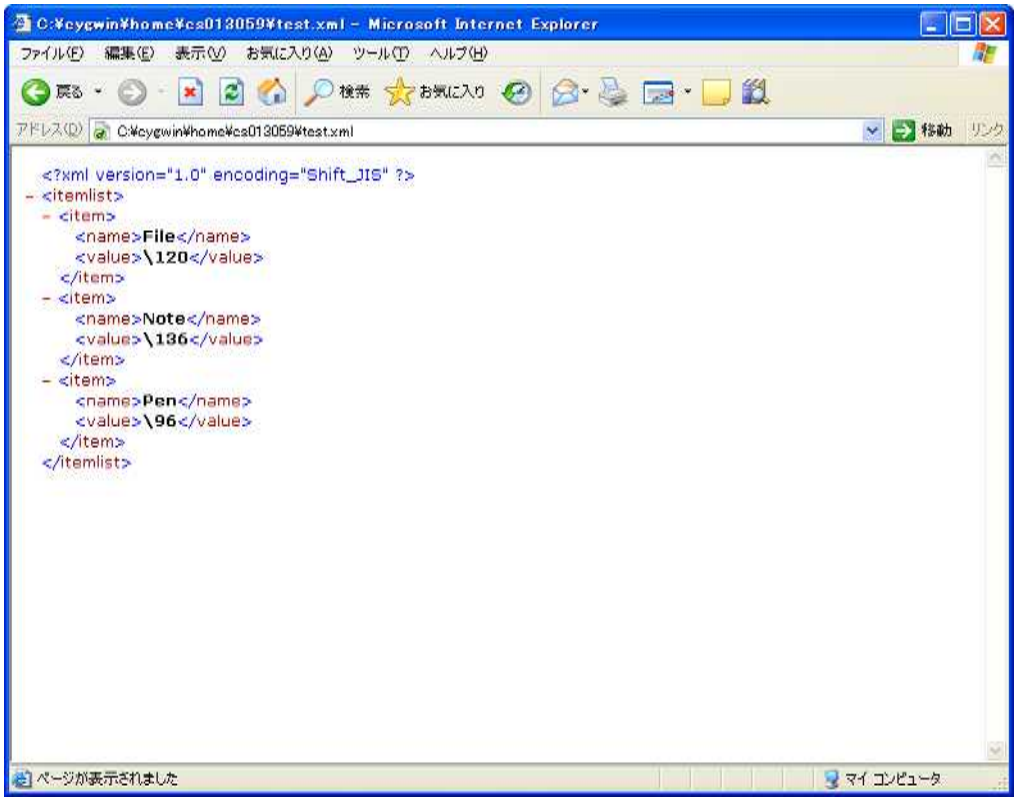
```
import javax.xml.parsers.*;
import javax.xml.transform.*;
import javax.xml.transform.dom.*;
import javax.xml.transform.stream.*;
import org.w3c.dom.*;

public class MakeDocument {
    public static void main ( String[] args ) {
        try {
            DocumentBuilderFactory factory = DocumentBuilderFactory.newInstance();
            DocumentBuilder builder = factory.newDocumentBuilder();

            // 空の DOM 木を生成
            Document doc = builder.newDocument();
            // DOM 木の操作 (空の木にノードを追加していく)
            Element root = doc.createElement ( "itemlist" );
            doc.appendChild ( root );

            // 一つ目の商品を追加
            Element item_01 = doc.createElement ( "item" );
            root.appendChild ( item_01 );
            Element name_01 = doc.createElement ( "name" );
            name_01.appendChild ( doc.createTextNode ( "File" ) );
            item_01.appendChild ( name_01 );
            Element val_01 = doc.createElement ( "value" );
            val_01.appendChild ( doc.createTextNode ( "￥￥120" ) );
            item_01.appendChild ( val_01 );

            // 二つ目の商品を追加
            Element item_02 = doc.createElement ( "item" );
            root.appendChild ( item_02 );
            Element name_02 = doc.createElement ( "name" );
            name_02.appendChild ( doc.createTextNode ( "Note" ) );
            item_02.appendChild ( name_02 );
            Element val_02 = doc.createElement ( "value" );
            val_02.appendChild ( doc.createTextNode ( "￥￥136" ) );
            item_02.appendChild ( val_02 );

            // 三つ目の商品を追加
            Element item_03 = doc.createElement ( "item" );
            root.appendChild ( item_03 );
            Element name_03 = doc.createElement ( "name" );
            name_03.appendChild ( doc.createTextNode ( "Pen" ) );
            item_03.appendChild ( name_03 );
            Element val_03 = doc.createElement ( "value" );
            val_03.appendChild ( doc.createTextNode ( "￥￥96" ) );
            item_03.appendChild ( val_03 );

            // 木構造を一次元に変換する (入力：DOMSource、出力：System.out)
            TransformerFactory tFactory = TransformerFactory.newInstance();
            Transformer tf = tFactory.newTransformer();
            // 文字コードの設定
            tf.setOutputProperty ( OutputKeys.ENCODING, "Shift_JIS" );
            DOMSource source = new DOMSource ( doc );
            StreamResult result = new StreamResult ( System.out );
            tf.transform ( source,result );
        } catch ( Exception e ) {
            e.printStackTrace();
        }
    }
}
```

【実行結果】

≪標準出力≫



≪ブラウザ出力≫



【考察】

標準出力では、改行がなされていないので見づらいが、正確に XML が作成されている。

ブラウザ出力では、XML 文書の生成と出力が正確に行われていることがより明確にわかる。