2006/02/08

平成17年度(2005年度) 計算機言語II定期試験

1. 説明問題(40点)
   Java言語に関する次の用語について説明せよ。(各10点)
   (a) クラスメソッド
   (b) privateフィールド
   (c) import宣言
   (d) メソッドやコンストラクタのオーバロード
2. Javaプログラミングに関する問題(30点)
   (I) 1以上1000以下の整数のうち、3で割り切れるが7で割り切れないものをすべて出力するJavaプログラムを書け。1行に1個の整数を出力するようにすること。(10点)
   (II) 右のJavaプログラム(IntegerCompare.java)を実行した結果はどのようになるか。またその理由を述べよ。(10点)

```
public class IntegerCompare {
  public static void main(String[] args) {
    int i = 0;
    int j = 0;
    Integer ii = new Integer(i);
    Integer jj = new Integer(j);
    System.out.println(i == j);
    System.out.println(ii == jj);
    System.out.println(ii.equals(jj));
  }
}
```

   (III) シロイヌ宅急便社では、宅配便の荷物の運送料を計算するJavaプログラムを実装中である。将来いろいろな形の荷物に対応するため、荷物を表す次のようなインタフェースNimotsuを設計した。直方体の荷物について、このインタフェースを実装するJavaのクラスを書け。荷物の辺の長さは整数値であると仮定してよい。(10点)

```
public interface Nimotsu {
  int getTaiseki(); // 体積を求める
  int getMaxLength(); // 一番長い辺の長さを求める
}
```

3. XMLに関する問題(30点)

以下に示すのは、個人の住所データを表すXMLである。これについて、以下の問に答えよ。

```
<?xml version="1.0"?>
<ContactXML version="1.0" creator="http://www.foo.com/bar/meishi-app/1.0">
  <PersonName>
    <PersonNameItem>
    <FullName pronunciation="ヤマダ タロウ">山田 太郎</FullName>
    <FirstName pronunciation="タロウ">太郎</FirstName>
    <LastName pronunciation="ヤマダ">山田</LastName>
  </PersonNameItem>
  </PersonName>
  <Address>
    <AddressItem locationType="Office">
      <Country countryCode="JP">日本</Country>
      <Zip>123-4567</Zip>
      <AddressLine addressLineType="StateOrPrefecture">東京都</AddressLine>
      <AddressLine addressLineType="City">品川区</AddressLine>
      <AddressLine addressLineType="Town">大井町</AddressLine>
      <AddressLine addressLineType="Number">1-2-3</AddressLine>
      <AddressLine addressLineType="Building">NT ビル</AddressLine>
      <AddressLine addressLineType="FloorOrRoom">10F</AddressLine>
    </AddressItem>
  </Address>
  <Occupation>
    <OccupationItem>
      <OrganizationName>ABC ソフト株式会社</OrganizationName>
      <Department>マーケティング部プロモーション課</Department>
      <JobTitle>係長</JobTitle>
    </OccupationItem>
  </Occupation>
  <Phone>
    <PhoneItem phoneDevice="Phone" usage="Official">+81-3-1234-5678</PhoneItem>
    <PhoneItem phoneDevice="Cellular" usage="Private">090-8765-4321</PhoneItem>
  </Phone>
  <Email>
    <EmailItem emailDevice="PC" usage="Official">aabbcc@abcd.com</EmailItem>
  </Email>
</ContactXML>
```

(I) このXMLにはいくつの要素が含まれているか。(5点)

(II) このXMLをDOMで表したとき、Textノードはいくつ含まれているか。(5点)

(III) 次に示すのは、SAXを利用してこのXMLに含まれる要素の個数を標準出力に表示するJavaプログラムである。「以下未完成」とある部分に入るメソッド(複数の場合あり)を実装せよ。(10点)

```
import java.io.IOException;

import javax.xml.parsers.ParserConfigurationException;
import javax.xml.parsers.SAXParser;
import javax.xml.parsers.SAXParserFactory;

import org.xml.sax.SAXException;
import org.xml.sax.helpers.DefaultHandler;

public class ElementCounter { // このクラスは完成している
	public static void main(String[] args) {
		SAXParserFactory factory = SAXParserFactory.newInstance();
		try {
			SAXParser parser = factory.newSAXParser();
			DefaultHandler handler = new ElementCounterHandler();
			parser.parse(args[0], handler);
		} catch (ParserConfigurationException e) {
			e.printStackTrace();
		} catch (SAXException e) {
			e.printStackTrace();
		} catch (IOException e) {
			e.printStackTrace();
		}
	}
}
```

```
import org.xml.sax.Attributes;
import org.xml.sax.helpers.DefaultHandler;

public class ElementCounterHandler extends DefaultHandler {
	private int count = 0;
	public void startDocument() {}
	public void endDocument() {
		System.out.println(count);
	}
	// 以下未完成
}
```

(IV) 次に示すのは、このXMLに対応するDOMに含まれるTextノードの個数を標準出力に表示するJavaプログラムである。「未完成部分その１」「未完成部分その２」「未完成部分その３」を適切に実装せよ。(10点)

```
import java.io.IOException;

import javax.xml.parsers.DocumentBuilder;
import javax.xml.parsers.DocumentBuilderFactory;
import javax.xml.parsers.ParserConfigurationException;

import org.w3c.dom.Node;
import org.w3c.dom.NodeList;
import org.xml.sax.SAXException;

public class TextNodeCounter {
	private int count = 0;
	public static void main(String[] args) {
		DocumentBuilderFactory factory =
			DocumentBuilderFactory.newInstance();
		try {
			DocumentBuilder builder = factory.newDocumentBuilder();
			Node root = builder.parse(args[0]);
			TextNodeCounter c = new TextNodeCounter();
			c.calc(root);
			System.out.println(c.count);
		} catch (ParserConfigurationException e) {
			e.printStackTrace();
		} catch (SAXException e) {
			e.printStackTrace();
		} catch (IOException e) {
			e.printStackTrace();
		}

	}
	public void calc(Node root) {
		switch (root.getNodeType()) {
		case Node.DOCUMENT_NODE:
			// 未完成部分その１
		case Node.ELEMENT_NODE:
			// 未完成部分その２
		case Node.TEXT_NODE:
			// 未完成部分その３
		default:
			break;
		}
	}
}
```