

Gruppe A	PRÜFUNG AUS "SEMISTRUKTURIERTE DATEN" 181.135		19. 06. 2009
Kennnr.	Matrikelnr.	Familienname	Vorname

Arbeitszeit: 120 Minuten. Aufgaben sind auf den Angabeblättern zu lösen; Zusatzblätter werden nicht gewertet. Bitte tragen Sie Namen, Studienkennzahl und Matrikelnummer ein. Halten Sie Ihren Studentenausweis bereit.

### Aufgabe 1:

(12)

Betrachten Sie die folgende XML-Schema Datei **test.xsd**:

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema">
  <xsd:element name="A" type="Atype"/>
  <xsd:complexType name="Atype">
    <xsd:choice minOccurs="0">
      <xsd:element name="A" type="Atype"/>
      <xsd:sequence minOccurs="2" maxOccurs="2">
        <xsd:element name="B" type="xsd:string"/>
      </xsd:sequence>
      <xsd:element name="C">
        <xsd:complexType mixed="true">
          <xsd:sequence>
            <xsd:element name="D" maxOccurs="unbounded" type="xsd:string"/>
            <xsd:element name="E" minOccurs="0" type="xsd:string"/>
          </xsd:sequence>
        </xsd:complexType>
      </xsd:element>
    </xsd:choice>
  </xsd:complexType>
</xsd:schema>
```

Betrachten Sie weiters die acht verschiedenen XML-Dateien, die unten angeführt sind.

Sie können davon ausgehen, dass alle folgenden XML-Dateien wohlgeformt sind. Es geht also lediglich darum, ihre Gültigkeit bezüglich **test.xsd** zu entscheiden.

Kreuzen Sie an, welche der folgenden XML-Dateien gültig bezüglich **test.xsd** sind.

- |   |  |  |
|---|--|--|
| 1. <A></A>  | gültig <input checked="" type="checkbox"/> | ungültig <input type="checkbox"/>            |
| 2. <A><B>abc</B></A>                                      | gültig <input type="checkbox"/>            | ungültig <input checked="" type="checkbox"/> |
| 3. <A><B>abc</B><B>def</B><B>ghi</B><B>jkl</B></A>        | gültig <input type="checkbox"/>            | ungültig <input checked="" type="checkbox"/> |
| 4. <A><C>abc<D>ghi</D><E>def</E>jkl</C></A>               | gültig <input checked="" type="checkbox"/> | ungültig <input type="checkbox"/>            |
| 5. <A><A><B>abc</B><B>def</B></A></A>                     | gültig <input checked="" type="checkbox"/> | ungültig <input type="checkbox"/>            |
| 6. <A><A></A><A></A></A>                                  | gültig <input type="checkbox"/>            | ungültig <input checked="" type="checkbox"/> |
| 7. <A><A><B>abc</B><B>def</B></A><B>ghi</B><B>jkl</B></A> | gültig <input type="checkbox"/>            | ungültig <input checked="" type="checkbox"/> |
| 8. <A><C>abc</C></A>                                      | gültig <input type="checkbox"/>            | ungültig <input checked="" type="checkbox"/> |

(Pro korrekter Antwort 1.5 Punkte, **pro inkorrektter Antwort -1.5 Punkte**, pro nicht beantworteter Frage 0 Punkte, für die gesamte Aufgabe mindestens 0 Punkte)

## Aufgabe 2:

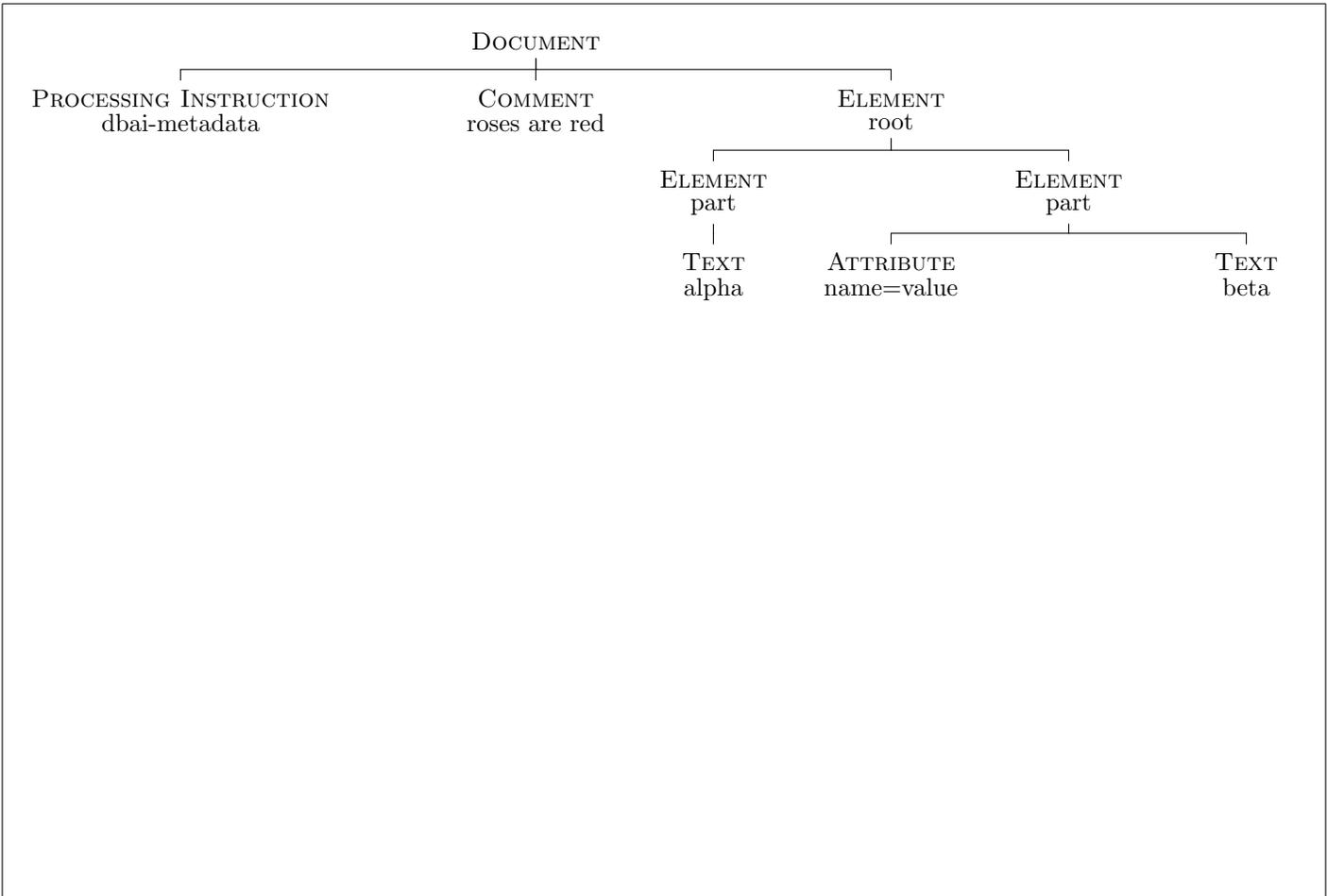
(8)

Nehmen Sie an, dass folgender Java Code (links) auf das angegebene XML-Dokument (rechts) ausgeführt wird. Zeichnen Sie den resultierenden DOM-Baum *nach* der Ausführung des Codes auf.

Bemerkung: Die Einrückungen im XML Dokument (Whitespaces) dienen nur der Darstellung und sollen *nicht* als Textknoten im Baum aufscheinen. `documentNode` ist die Wurzel des XML Dokuments, `rootNode` ist somit eine Referenz auf das `root` Element im XML Dokument.

```
class Exam {
    public static void main() {
        Document documentNode; // DOCUMENT Node
        Node rootNode; // Wurzelement (root)
        // Gegeben: Dokument lesen etc...
        rootNode = documentNode.getDocumentElement();
        something(rootNode);
        // Aufgabe: DOM Baum zeichnen
    }
    private static void something(Node n) {
        NodeList nl = n.getChildNodes();
        for(int i=nl.getLength()-1; i >= 2; --i) {
            Node child = nl.item(i);
            n.removeChild(child);
        }
    }
}
```

```
<?xml version="1.0"?>
<?dbai-metadata?>
<!--roses are red-->
<root>
  <part>alpha</part>
  <part name="value">beta</part>
  <part>
    <subpart>omega</subpart>
    <subpart>psi</subpart>
    <subpart>chi</subpart>
  </part>
  <part>delta</part>
  <part>epsilon</part>
</root>
```



### Aufgabe 3:

(6)

Betrachten Sie die folgende XML- Datei **ns.xml**:

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<wurzel xmlns="http://uri1.at"
        xmlns:myprefix="http://uri2.at">
  <erstes attr="wert"/>
  <myprefix:zweites xmlns="http://uri3.at">
    <drittes/>
  </myprefix:zweites>
</wurzel>
```

Kreuzen Sie an, ob die folgenden Aussagen für die Datei **ns.xml** wahr oder falsch sind.

1. Das Element **erstes** liegt im Namespace **http://uri1.at**. wahr  falsch
2. Das Attribut **attr** liegt im Namespace **http://uri1.at**. wahr  falsch
3. Das Element **myprefix:zweites** liegt im Namespace **http://uri3.at**. wahr  falsch
4. Das Element **drittes** liegt im Namespace **http://uri3.at**. wahr  falsch

### Aufgabe 4:

(9)

Kreuzen Sie an, ob die folgenden Aussagen wahr oder falsch sind.

1. Wenn ein XML Dokument nicht wohlgeformt ist, muss der Parser die Verarbeitung abbrechen. wahr  falsch
2. `<der:text-eins_zwei>text</der:text-eins_zwei>` ist wohlgeformt. wahr  falsch
3. Folgende zwei Ausdrücke sind äquivalent (d.h. sie liefern immer dasselbe Resultat):  
`//paragraph[1][last()]`  
`//paragraph[last()][1]` wahr  falsch
4. Der XPath-Ausdruck `./a[@b=c]` ist die Kurzschreibweise des XPath-Ausdrucks  
`self::node()/child::a/attribute::b[c]` wahr  falsch
5. Um die Wohlgeformtheit eines XML-Dokuments zu überprüfen, muss eine DTD oder ein XML-Schema vorhanden sein. wahr  falsch
6. Der Speicherbedarf eines DOM-Parsers ist abhängig von der Größe des XML-Dokuments. wahr  falsch

(Pro korrekter Antwort 1.5 Punkte, **pro inkorrektter Antwort -1.5 Punkte**, pro nicht beantworteter Frage 0 Punkte, für die gesamte Aufgabe mindestens 0 Punkte)

**Bemerkung:** Die restlichen Beispiele (Aufgaben 5–8) beziehen sich alle auf XML-Dokumente der Gestalt **verzeichnis.xml** (siehe letzte Seite — Sie können die letzte Seite auch abtrennen)!

**Aufgabe 5:**

(12)

Geben Sie eine DTD **verzeichnis.dtd** an, sodass das XML-Dokument **verzeichnis.xml** (siehe Anhang) bezüglich dieser DTD gültig ist. Berücksichtigen Sie beim Erstellen der DTD folgende Punkte:

- Ein **verzeichnis** besteht aus einer beliebigen Anordnung von Elementen **verzeichnis**, **datei**, und **verweis**. Ein Verzeichnis kann auch leer sein! Beachten Sie weiters, dass Verzeichnisse beliebig tief verschachtelt sein können.
- Elemente **verzeichnis** und **datei** haben ein verpflichtendes Attribut **name**. Element **verweis** hat verpflichtende Attribute **vname** und **dname**, aber keinen Elementinhalt. Das Attribut **position** soll, falls nicht angegeben, einen Defaultwert "1" erhalten. Die Attributwerte sollen (bzgl. des Typs) keinen Einschränkungen unterliegen.
- Ein **datei** Element hat eine beliebige Anzahl von **log**-Kindelementen. **log**-Elemente haben gemischten Inhalt und beinhalten eine beliebige Kombination von Elementen **vname** und **dname**, die ihrerseits beliebigen Textinhalt haben.

Datei **verzeichnis.dtd**:

```
<!ELEMENT verzeichnis (verzeichnis|datei|verweis)*>
<!ATTLIST verzeichnis name CDATA #REQUIRED>
<!ELEMENT datei (log)*>
<!ATTLIST datei name CDATA #REQUIRED>
<!ELEMENT log (#PCDATA|vname|dname)*>
<!ELEMENT verweis EMPTY>
<!ATTLIST verweis vname CDATA #REQUIRED>
<!ATTLIST verweis dname CDATA #REQUIRED>
<!ATTLIST verweis position CDATA "1">
<!ELEMENT vname (#PCDATA)>
<!ELEMENT dname (#PCDATA)>
```

### Aufgabe 6:

(12)

Betrachten Sie die folgenden drei XSLT-Stylesheets. Geben Sie jeweils den Output an, den das entsprechende Stylesheet angewandt auf **verzeichnis.xml** (siehe Anhang) liefert. Sie brauchen sich dabei nicht um Whitespaces etc. kümmern.

Anmerkung: Pro Teilaufgabe sind jeweils 4 Punkte erreichbar.

Datei **query1.xsl**:

```
<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
<xsl:output method="xml" indent="yes" encoding="UTF-8" version="1.0" />

<xsl:template match="/">
  <xsl:for-each select="//verweis">
    <xsl:copy-of select="."/>
  </xsl:for-each>
</xsl:template>

</xsl:stylesheet>
```

Geben Sie hier den Output von **query1.xsl** angewandt auf **verzeichnis.xml** an:

```
<verweis vname="myxquery" dname="xquery" position="1"/>
<verweis vname="myxpathtester" dname="xpathtester" position="2"/>
```

Datei **query2.xsl**:

```
<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
<xsl:output method="xml" indent="yes" encoding="UTF-8" version="1.0" />

<xsl:template match="/">
  <xsl:apply-templates select="//datei[@name='xpathtester']"/>
</xsl:template>

<xsl:template match="//datei">
  <xsl:text>Datei gefunden!</xsl:text>
</xsl:template>

<xsl:template match="//verweis">
  <xsl:text>Verweis gefunden!</xsl:text>
</xsl:template>

<xsl:template match="log">
</xsl:template>

</xsl:stylesheet>
```

Geben Sie hier den Output von **query2.xsl** angewandt auf **verzeichnis.xml** an:

```
Datei gefunden!
Datei gefunden!
```

Datei **query3.xsl**:

```
<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
<xsl:output method="xml" indent="yes" encoding="UTF-8" version="1.0" />

<xsl:template match="/">
  <xsl:apply-templates/>
</xsl:template>

<xsl:template match="//datei[@name='xpathtester']">
  <xsl:text>Datei gefunden!</xsl:text>
</xsl:template>

<xsl:template match="//verweis">
  <xsl:text>Verweis gefunden!</xsl:text>
</xsl:template>

<xsl:template match="log">
</xsl:template>

</xsl:stylesheet>
```

Geben Sie hier den Output von **query3.xsl** angewandt auf **verzeichnis.xml** an:

```
Datei gefunden!
Verweis gefunden!
Datei gefunden!
Verweis gefunden!
```

### Aufgabe 7:

(8)

Schreiben Sie einen **XMLFilter** der alle **dname** und **vname**-Elemente entfernt. Der enthaltene Text soll erhalten bleiben. Sie müssen also die Events für *startElement* und *endElement* entfernen.

Beispiel (Input → Output):

```
<verzeichnis name="bin">
  <datei name="xquery"/>
  <datei name="xpathtester">
    <log>Erstellt am 01.01.2009</log>
    <log>Umbenannt von <dname>xpathviewer</dname>
      in <dname>xpathtester</dname></log>
  </datei>
</verzeichnis>
```

```
<verzeichnis name="bin">
  <datei name="xquery"/>
  <datei name="xpathtester">
    <log>Erstellt am 01.01.2009</log>
    <log>Umbenannt von xpathviewer
      in xpathtester</log>
  </datei>
</verzeichnis>
```

Tipp: Die Klasse `LogFilter` erweitert `XMLFilterImpl`, dh. per default werden alle Events durchgereicht. Überschreiben Sie die relevanten Methoden und geben Sie ein Event nur dann an die Superklasse weiter (mittels `super.methode(...parameter...)`) wenn Sie es behalten wollen.

```
class LogFilter extends XMLFilterImpl {

    public void startElement(String namespaceURI, String localName, String qName, Attributes atts)
    throws SAXException {
        if(!localName.equals("dname") && !localName.equals("vname")) {
            super.startElement(namespaceURI, localName, qName, atts);
        }
    }

    public void endElement(String namespaceURI, String localName, String qName)
    throws SAXException {
        if(!localName.equals("dname") && !localName.equals("vname")) {
            super.endElement(namespaceURI, localName, qName);
        }
    }
}
```

## Aufgabe 8:

(8)

Schreiben Sie XPath-Anfragen für folgende Aufgabenstellungen. Zu jeder Abfrage ist ein Beispiel mit der erwarteten Ausgabe (bezgl. `verzeichnis.xml`) angegeben. Die Abfragen sollen auf allen Dokumenten, die gültig bezgl. `verzeichnis.dtd` sind, funktionieren.

1. Geben Sie alle Verzeichnisnamen aus.

```
/  
bin  
usr  
...
```

```
//verzeichnis/@name
```

2. Geben Sie den Namen aller Dateien aus, für die mindestens ein `log`-Eintrag vorhanden ist.

```
xpathtester  
xquery
```

```
//datei[./log]/@name
```

3. Geben Sie für jedes Verzeichnis den Namen und die Anzahl der direkt enthaltenen Dateien aus.

```
/ 0  
bin 2  
usr 0  
...
```

```
//verzeichnis/concat(@name, ' ', count(datei))
```

4. Geben Sie jene `verzeichnis` Elemente aus, die im gesamten Unterbaum kein `verweis` Element enthalten. Beispiel:

```
<verzeichnis name="bin">...</verzeichnis>  
<verzeichnis name="src">...</verzeichnis>
```

```
//verzeichnis[not(./verweis)]
```

## Sie können diese Seite abtrennen!

Das folgende XML-Dokument `verzeichnis.xml` gilt für Aufgaben 5–8:

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE verzeichnis SYSTEM "verzeichnis.dtd">
<verzeichnis name="/">
  <verzeichnis name="bin">
    <datei name="xquery"/>
    <datei name="xpathtester">
      <log>Erstellt am 01.01.2009</log>
      <log>Umbenannt von <dname>xpathviewer</dname>
        in <dname>xpathtester</dname></log>
    </datei>
  </verzeichnis>
<verzeichnis name="usr">
  <verzeichnis name="jakl">
    <verzeichnis name="bin">
      <verweis vname="myxquery" dname="xquery" position="1"/>
    </verzeichnis>
    <datei name="xpathtester"/>
  </verzeichnis>
<verzeichnis name="woltran">
  <verzeichnis name="bin">
    <verweis vname="myxpathtester" dname="xpathtester" position="2"/>
    <datei name="xquery">
      <log>Kopiert aus Verzeichnis <vname>/bin</vname></log>
    </datei>
  </verzeichnis>
  <datei name="info.txt"/>
  <verzeichnis name="src">
    <datei name="example1.xml"/>
  </verzeichnis>
</verzeichnis>
</verzeichnis>
</verzeichnis>
```

Gesamtpunkte: 75