Klausur: Programmierkurs 1, Java, XML

O 163, Schloss

6. März 2002

Name:		Vorname:	
Matrikelnummer:	Studiengang:	Lfd.Nr.:	Sitz-Nr.:

Hinweise zur Bearbeitung

a. Die Klausur besteht aus **5 Aufgaben**, für die insgesamt **?? Punkte** vergeben werden. Die Klausur gilt als bestanden, wenn davon **mindestens ?? Punkte** (d.h. 50 %) erreicht werden.

b. Eigenes Papier und sonstige Hilfsmittel (z.B. Taschenrechner, Palm, Notebook, Handy) sind nicht zugelassen. Die Rückseiten der Klausurbögen dürfen verwendet werden.

c. Alle Lösungswege und Rechengänge sind grundsätzlich zu dokumentieren. **Lösungen ohne Lösungsweg werden nicht gewertet.**

Viel Erfolg !

Unterschrift des Kandidaten:

Bitte nicht ausfüllen:

Aufgabe	1	2	3	4	5	Summe
Punkte						

Bestanden	Korrektor
ja | nein	

Aufgabe 1 (12 Punkte)

a. Schreiben Sie ein HTML-Formular (keine vollständige Seite) entsprechend den folgenden Vorgaben:

- Nach Submitierung des Formulars soll das CGI-Programm `http://www.uni-mannheim.de/formular.cgi` ausgeführt werden.
- Es soll zwei Texteingabefelder mit der Bezeichnung '`Vorname`' bzw. '`Familienname`' geben. Diese Felder sollen den Namen '`given`' bzw. '`family`' haben und jeweils Platz für 30 Zeichen in der Anzeige haben. Die Felder sollen in zwei Zeilen erscheinen.
- Der Submit-Button soll mit '`Abschicken`' beschriftet sein.

(6 Punkte)

b. Schreiben Sie das HTML-Formular XML-konfomrm um, d.h., schreiben Sie es als XHTML-Formular. (2 Punkte)

c. Geben Sie CSS-Regeln für folgende Elemente und Darstellungen an:

- Ein `em` innerhalb von `p` soll mit blauer, kursiver Schrift dargestellt werden.
- Ein `em` innerhalb von `strong` soll mit roter, fetter Schrift dargestellt werden.

(4 Punkte)

Aufgabe 2 (11 Punkte)

a. Was versteht man unter Variablen? (2 Punkte)

b. Wodurch unterscheiden sich Variablen elementaren Typs und Variablen eines Referenztypes? (3 Punkte)

c. **Überladen** Sie die Methode `summe` für den Datentyp `double` in der Klasse `X`.

```
public class X {
 public static long summe(long a, long b) { return a+b; }
}
```

(3 Punkte)

d. **Überschreiben** Sie die Methode `summe` mit einer, die den 'Algorithmus' `-(-a-b)` zur Berechnung verwendet. (3 Punkte)

Aufgabe 3 (14 Punkte)

Die folgende Klasse `Zufall` speichert Zufallszahlen.

```
public class Zufall {

    private double[] zahlen = null;


    public Zufall(int anzahl) {









    }


    public ... () {











    }


    public static void main(String[] args) {
        Zufall zahlen = new Zufall(5);
        System.out.println("Zufallszahlen:\n" + zahlen);
    }
}
```

a. Implementieren Sie den Konstruktor, so dass er die gewünschte Anzahl von Zufallszahlen in der Variablen `zahlen` erzeugt. Prüfen Sie, dass `anzahl` größer oder gleich null ist. (7 Punkte)

b. Implementieren Sie eine weitere Methode ..., so dass die Ausgabe von `main` wie folgt sein kann:

```
Zufallszahlen:
Zahl 0 = 0.33186561845760343
Zahl 1 = 0.6772109033787419
Zahl 2 = 0.9112870102890398
Zahl 3 = 0.44918730813656516
Zahl 4 = 0.4094348315175249
```

(7 Punkte)

Aufgabe 4 (16 Punkte)

Die folgende Klasse `Counter` speichert Zählerstande für verschiedene Zähler.

```
import java.util.Hashtable;

public class Counter {

    private Hashtable db = null;

    public Counter() {


    }

    public long count(String key) {











    }

}
```

Das folgende Hauptprogramm

```
    public static void main(String[] args) {
        Screen out = new Screen();
        String de = "Deutschland";
        String fr = "Fankreich";
        Counter zaehler = new Counter();
        out.print(" de = " + zaehler.count(de) );
        out.print(" de = " + zaehler.count(de) );
```

```
        out.print(" fr = " + zaehler.count(fr) );
        ...
    }
```

soll diese Ausgabe erzeugen: `de = 1 de = 2 fr = 1 ...`

Verwenden Sie die Methoden `get` und `put` von `Hashtable`.

```
/**
* Returns the value to which the specified key is
* mapped in this hashtable.
* @param   key   a key in the hashtable.
* @return  the value to which the key is mapped in this
*          hashtable; <code>null</code> if the key is not
*          mapped to any value in this hashtable.
*/
    public synchronized Object get(Object key)

/**
* Maps the specified <code>key</code> to the specified
* <code>value</code> in this hashtable. Neither the key
* nor the value can be <code>null</code>. <p>
*
* The value can be retrieved by calling the <code>get</code>
* method with a key that is equal to the original key.
*
* @param      key     the hashtable key.
* @param      value   the value.
* @return     the previous value of the specified key
*             in this hashtable, or <code>null</code> if
*             it did not have one.
* @exception  NullPointerException  if the key or value is
*             <code>null</code>.
*/
    public synchronized Object put(Object key, Object value)
```

a. Implementieren Sie den Konstruktor. (2 Punkte)

b. Implementieren Sie die Methode `count`. Ein Aufruf von
`count('NameDesZaehlers')`
soll den Zählerstand um `1` erhöhen und zurückliefern. (10 Punkte)

c. Implementieren Sie den Konstruktor `SAXZaehler` und die Methode `startElement` eines `SAXHandlers`, die die verschiedenen Elemente zählt. Verwenden Sie dazu die Klasse `Counter`.

```
public class SAXZaehler extends DefaultHandler {

     Counter zaehler = null;

     public SAXZaehler() {


     }

    public void startElement(String nameuri,
                             String local,
                             String quali,
                             Attributes attrs) {








    }
}
```

(4 Punkte)

Aufgabe 5 (13 Punkte)

Gegeben sei folgende XML-DTD:

```
<?xml version="1.0" encoding="iso-8859-1" ?>
<!ELEMENT a ( (c|b)*, d+, e? ) >
<!ELEMENT b EMPTY >
<!ELEMENT c EMPTY >
<!ELEMENT d EMPTY >
<!ELEMENT e EMPTY >
<!ATTLIST b id ID #REQUIRED >
<!ATTLIST c name NMTOKEN #IMPLIED >
<!ATTLIST d lang (en|gr|fr) "en" >
<!ATTLIST e sign CDATA #FIXED "hk" >
```

Welche der folgenden Ausschnitte sind 'wohlgeformt' (well formed) und welche sind 'gültig' (valid) (mit Begründung):

a. `<a> <d /> </a>` (1 Punkt)

b. `<a> <e /> </a>` (1 Punkt)

c. `<a> <d> </a>` (1 Punkt)

d. `<a> <b id="n1" /> <c> <b id="n1" /> <d /> </a>` (2 Punkte)

e. `<a> <b id="n1" /> <c /> <b id="n1" /> <d />`
`</a>` (2 Punkte)

f. `<a> <c /> <b id=n2 /> <d /> <e sign="heinz" />`
`</a>` (2 Punkte)

g. `<a> <b id="n1" /> <c /> <d lang="tr" /> <e />`
`</a>` (2 Punkte)

h. `<a> <b id="n1" /> <c name="t1" /> <c name="t1" />`
`<d lang="gr" /> <e sign="hk" /> </a>` (2 Punkte)