

Vorname

Nachname

Matrikel-Nr

Diese Klausur ist mein **letzter Prüfungsversuch** (bitte ankreuzen): Ja ☐ Nein ☐

Schreiben Sie jede Lösung auf die Vorderseite eines *neuen Blattes* (und lassen Sie die Rückseiten Ihrer Lösungsblätter *leer*). Die Aufgaben 5 und 6 stehen auf der Rückseite dieses Blatts!

Aufgabe 1 (20 Punkte): Schreiben Sie eine Methoden, die der folgenden Spezifikation entspricht:

```
1  static public int anzX(String s, int i) {
2      // Wie oft kommt 'X' in s ab der Position i vor? Diese Funktion
3      // liefert die Antwort. Beispiele:
4      // anzX("AXBXCXDX", 2) ist gleich 3
5      // anzX("AXBXCXDX", 0) ist gleich 4
6      // anzX("AXBXCXDX", 8) ist gleich 0
7      // anzX("X"          , 0) ist gleich 1
8      // anzX("X"          , 1) ist gleich 0
9      // anzX("A"          , 0) ist gleich 0
10     // anzX(" "          , 0) ist gleich 0
11     //
12     // Ihre Loesung muss rekursiv sein (Schleifen sind im Rumpf dieser
13     // Funktion nicht erlaubt).
14     ...
15 } // anzX
```

Aufgabe 2 (20 Punkte): Schreiben Sie eine Funktion, die der folgenden Spezifikation entspricht:

```
1  static public boolean enthaeltMethode(String kn, String mn, int ap) {
2      // Liefert true genau dann wenn es eine Klasse namens kn gibt und
3      // in dieser Klasse eine Methode namens mn vereinbart wurde, die
4      // genau ap viele Parameter erwartet. Erlaeuterungen zu den Namen:
5      // "kn" wie Klassen-Name,
6      // "mn" wie Methoden-Name und
7      // "ap" wie Anzahl Parameter.
8      ...
9  } // enthaeltMethode
```

Aufgabe 3 (15 Punkte): Betrachten Sie die folgende Klassenvereinbarung:

```
1  static class Knoten {
2      Knoten next;
3      String data;
4
5      Knoten(Knoten next, String data) {
6          this.next = next;
7          this.data = data;
8      }
9  } // class Knoten
```

In einem Programm wird die Klasse Knoten wie folgt benutzt:

```
10     ...
11     Knoten kc = new Knoten(null, "KC");
12     Knoten kb = new Knoten(kc, "KB");
13     Knoten ka = new Knoten(kb, "KA");
14     ...
15     kc.next = kb;
16     kb.next = ka;
17     ka.next = kc;
18     ...
```

Wie sehen die Variablen ka, kb und kc (als Bojen dargestellt) aus, wenn der Ausführer

3.1. die Zeile 14 erreicht?

3.2. die Zeile 18 erreicht?

Aufgabe 4 (15 Punkte): Betrachten Sie die folgende Dokumenten Typ Definition (DTD):

```

1 <?xml version="1.0" encoding="UTF-8"?>
2 <!ELEMENT buch      (autor+, titel, verlag?)>
3 <!ELEMENT autor      (#PCDATA)>
4 <!ELEMENT titel       (#PCDATA)>
5 <!ELEMENT verlag      (#PCDATA)>
6 <!ELEMENT edition     EMPTY>
7 <!ATTLIST buch
8   edition      (1.|2.|3.|4.|5.|XX) "1."
9   sprache      (DE|ES|FR|EN|XX)      #IMPLIED
10  preis        CDATA                  #REQUIRED
11  waehrung      (Dollar|Euro|DM|XX)   #IMPLIED
12  isbn         CDATA                  #REQUIRED
13 >

```

Welche der folgenden fünf XML-Dokumente sind entsprechend dieser DTD *gültig* und welche nicht? Beschreiben Sie für die nicht-gültigen Dokumente jeweils kurz, *warum* sie nicht gültig sind.

Dokument BuchA:

```

14 <?xml version="1.0" encoding="UTF-8"?>
15 <!DOCTYPE anschrift SYSTEM "Buch.dtd">
16 <buch preis="25.00" isbn="0-123456789">
17   <autor>Joshua Bloch</autor>
18   <titel>Effective Java</titel>
19 </buch>

```

Dokument BuchB:

```

20 <?xml version="1.0" encoding="UTF-8"?>
21 <!DOCTYPE anschrift SYSTEM "Buch.dtd">
22 <buch edition="3." sprache="EN" preis="49.99" isbn="0-2345678901"
23   waehrung="Dollar">
24   <autor>James Gosling</autor>
25   <autor>Bill Joy</autor>
26   <autor>Guy Steele</autor>
27   <autor>Gilad Bracha</autor>
28   <titel>The Java Language Specification</titel>
29   <titel>A Formal Definition</titel>
30 </buch>

```

Dokument BuchC:

```

31 <?xml version="1.0" encoding="UTF-8"?>
32 <!DOCTYPE anschrift SYSTEM "Buch.dtd">
33 <buch preis="65.00" waehrung="DM" sprache="ED" isbn="3-3456789012">
34   <autor>Elliotte Rusty Hardy</autor>
35   <autor>W. Scott Means</autor>
36   <titel>XML In A Nutshell</titel>
37 </buch>

```

Dokument BuchD:

```

38 <?xml version="1.0" encoding="UTF-8"?>
39 <!DOCTYPE anschrift SYSTEM "Buch.dtd">
40 <buch edition="2." waehrung="DM" preis="71" isbn="2-4567890123">
41   <autor>David Harel</autor>
42   <titel>Algorithmics</titel>
43 </buch>

```

Dokument BuchE:

```

44 <?xml version="1.0" encoding="UTF-8"?>
45 <!DOCTYPE anschrift SYSTEM "Buch.dtd">
46 <buch waehrung="Dollar" edition="6." preis="49.99" isbn="2-34-5678-901">
47   <autor>James Gosling</autor>
48   <autor>Bill Joy</autor>
49   <autor>Guy Steele</autor>
50   <autor>Gilad Bracha</autor>
51   <titel>The Java Language Specification</titel>
52 </buch>

```

Aufgabe 5: (15 Punkte)

Beantworten Sie die folgenden Fragen möglichst kurz, aber genau. Vermeiden Sie Ausführungen, die *nichts mit der Frage zu tun haben* (Fehler in solchen überflüssigen Ausführungen können zu einem Punktabzug führen).

5.1. In Java unterscheidet man zwei Arten von Fäden: *Benutzer-Fäden* und *Dämonen* (user threads and demons). Wann wird ein Dämon *beendet*?

5.2. Wie lange eine abstrakter Algorithmus zur Lösung eines Problems der Größe n braucht, wird nicht in Sekunden angegeben, sondern in *Schritten*. Was ist *ein Schritt*?

5.3. Wir haben uns mit verschiedenen Formen von Sammlungen befaßt. Welchen Vorteil haben *sortierte* (verkettete) Listen im Vergleich zu *unsortierten* (verketteten) Listen?

5.4. *Wie viele Knoten* kann ein binärer Baum der Tiefe 25 höchstens haben? Geben Sie *eine* (ungefähre) *Zahl* an, keine Formel wie 17^5 oder $8 * 5^2$, die man noch ausrechnen muss.

5.5. In der Java-Standardbibliothek gibt es einige *Sammlungsklassen*, deren Namen die Zeichenkette `List` enthält (z.B. die Klassen `ArrayList` und `ConcurrentLinkedList`). Was ist die charakterisierende Eigenschaft von `List`-Sammlungen?

5.6. Durch welche Methode in welcher Schnittstelle wird die *natürliche Ordnung* einer Klasse festgelegt?

Aufgabe 6 (15 Punkte): Geben Sie für jede der Methoden `zk01` bis `zk06` an, welche *Zeitkomplexität* sie hat (z.B. $O(n)$ oder $O(n^2)$ oder $O(n^3)$ oder $O(\log(n))$ oder $O(2^n)$ oder ...). Gehen Sie dabei davon aus, dass eine Ausführung der Methode `machWas` als *ein Schritt* gezählt werden kann.

```
1  static void zk01(int n) {
2      for (int i=0; i<3*n; i++) {
3          machWas();
4      }
5      for (int i=0; i<2*n; i++) {
6          machWas();
7      }
8  } // zk01
9
10 static void zk02(int n) {
11     for (int i=0; i<n*n; i++) {
12         machWas();
13     }
14     for (int i=0; i<n*n*n; i++) {
15         machWas();
16     }
17 } // zk02
18
19 static void zk03(int n) {
20     for (int i=0; i<3*n; i++) {
21         for (int j=0; j<2*n; j++) {
22             machWas();
23         }
24     }
25 } // zk03
26
27 static void zk04(int n) {
28     for (int i=0; i<n*n; i++) {
29         for (int j=0; j<n*n*n; j++) {
30             machWas();
31         }
32     }
33 } // zk04
34
```

```

35 static void zk05(int n) {
36     for (int i=0; i<n; i++) {
37         for (int j=0; j<3*n; j++) {
38             for (int k=0; k<n/2; k++) {
39                 machWas();
40             }
41         }
42     }
43 } // zk05
44
45 static void zk06(int n) {
46     for (int i=0; i<5; i++) {
47         for (int j=0; j<3*n; j++) {
48             for (int k=0; k<2; k++) {
49                 machWas();
50             }
51         }
52     }
53 } // zk06

```

Beurteilung dieser Klausur:

A1	
A2	
A3	
A4	
A5	
A6	
Summe	
Note	
Datum	

Korrigierte Beurteilung:

A1	
A2	
A3	
A4	
A5	
A6	
Summe	
Note	
Datum	

Lösung 1 (20 Punkte):

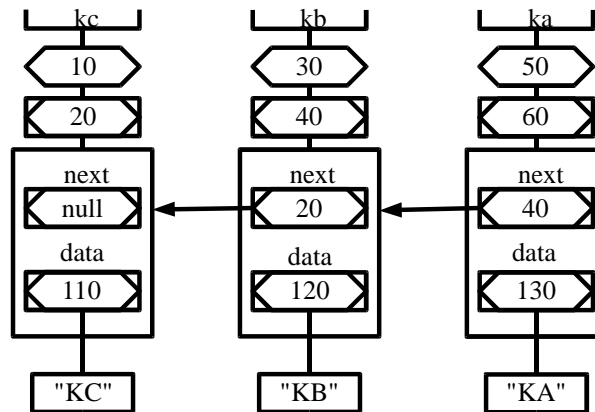
```
1  static public int anzX(String s, int i) {
2      // Wie oft kommt 'X' in s ab der Position i vor? Diese Funktion
3      // liefert die Antwort. Beispiele:
4      // anzX("AXBXCXDX", 2) ist gleich 3
5      // anzX("AXBXCXDX", 0) ist gleich 4
6      // anzX("AXBXCXDX", 8) ist gleich 0
7      // anzX("X"          , 0) ist gleich 1
8      // anzX("X"          , 1) ist gleich 0
9      // anzX("A"          , 0) ist gleich 0
10     // anzX("")          , 0) ist gleich 0
11     //
12     // Ihre Loesung muss rekursiv sein (Schleifen sind im Rumpf dieser
13     // Funktion nicht erlaubt).
14
15     if (i >= s.length()) return 0;
16     int imRest = anzX(s, i+1);
17     if (s.charAt(i) == 'X') {
18         return imRest + 1;
19     } else {
20         return imRest;
21     }
22 } // anzX
```

Lösung 2 (20 Punkte):

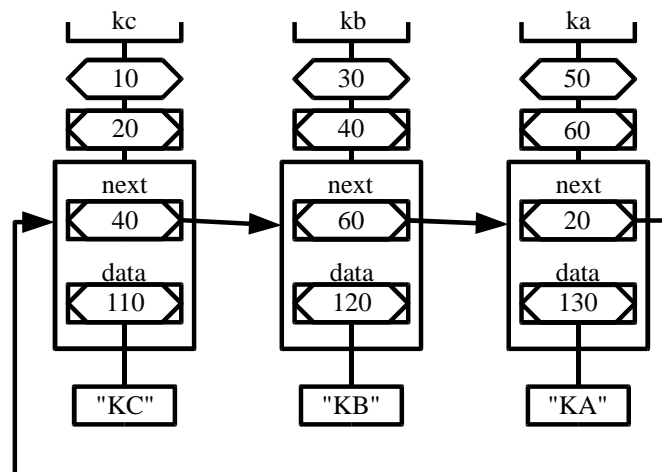
```
1  static public boolean enthaeltMethode(String kn, String mn, int ap) {
2      // Liefert true genau dann wenn es eine Klasse namens kn gibt und
3      // in dieser Klasse eine Methode namens mn vereinbart wurde, die
4      // genau ap viele Parameter erwartet.
5      // "kn" wie Klassen-Name,
6      // "mn" wie Methoden-Name und
7      // "ap" wie Anzahl Parameter.
8
9      try {
10         Class<?> kob = Class.forName(kn);
11         Method[] mr = kob.getDeclaredMethods();
12
13         for (Method m : mr) {
14             String mn2 = m.getName();
15             if (!mn.equals(mn2)) continue;
16             Class[] pts = m.getParameterTypes();
17             if (pts.length == ap) return true;
18         }
19         return false;
20     } catch (ClassNotFoundException ex) {
21         return false;
22     } catch (Exception ex) {
23         pln("Ausnahme in enthaeltMethode:");
24         pln(ex);
25     }
26
27     return true;
28 } // enthaeltMethode
```

Lösung 3 (15 Punkte):

Wie sehen die Variablen *ka*, *kb* und *kc* (als Bojen dargestellt) aus, wenn der Ausführer 3.1. die Zeile 14 erreicht?



Wie sehen die Variablen *ka*, *kb* und *kc* (als Bojen dargestellt) aus, wenn der Ausführer 3.2. die Zeile 18 erreicht?

**Lösung 4 (15 Punkte):**

```

1 ValidiereBuecher: Jetzt geht es los!
2 -----
3 istGueltig(BuchA.xml): true
4 -----
5 DOMError (error): The content of element type "buch"
6 must match "(autor+,titel,verlag?)".
7 istGueltig(BuchB.xml): false
8 -----
9 DOMError (error): Attribute "sprache" with value "ED"
10 must have a value from the list "DE ES FR EN XX ".
11 istGueltig(BuchC.xml): false
12 -----
13 istGueltig(BuchD.xml): true
14 -----
15 DOMError (error): Attribute "edition" with value "6."
16 must have a value from the list "1. 2. 3. 4. 5. XX ".
17 istGueltig(BuchE.xml): false
18 -----
19 ValidiereBuecher: Das war's erstmal!

```

Lösung 5 (15 Punkte):

5.1. In Java unterscheidet man zwei Arten von Fäden: *Benutzer-Fäden* und *Dämonen* (user threads and demons). Wann wird ein Dämon *beendet*?

Wenn alle Benutzer-Fäden (und damit das gesamte umgebende Programm) sich beenden.

5.2. Wie lange eine abstrakter Algorithmus zur Lösung eines Problems der Größe n braucht, wird nicht in Sekunden angegeben, sondern in *Schritten*. Was ist *ein Schritt*?

Eine Folge von Befehlen von der man annehmen kann, dass ein Ausführer sie immer in (ungefähr) der gleichen Zeit ausführen kann. Diese Zeit muss unabhängig von Größe n des zu lösenden Problems sein.

5.3. Wir haben uns mit verschiedenen Formen von Sammlungen befaßt. Welchen Vorteil haben *sortierte* (verkettete) Listen im Vergleich zu *unsortierten* (verketteten) Listen?

Beim Suchen im negativen Fall braucht man bei der unsortierten Liste immer n Schritte, bei der sortierten Liste im Durchschnitt aber nur $n/2$ Schritte.

5.4. *Wie viele Knoten* kann ein binärer Baum der Tiefe 25 höchstens haben? Geben Sie *eine* (ungefähre) *Zahl* an, keine Formel wie 17^5 oder $8 * 5^2$, die man noch ausrechnen muss.

Etwa 32 Millionen Knoten.

5.5. In der Java-Standardbibliothek gibt es einige *Sammlungsklassen*, deren Namen die Zeichenkette `List` enthält (z.B. die Klassen `ArrayList` und `ConcurrentLinkedList`). Was ist die charakterisierende Eigenschaft von `List`-Sammlungen?

Jede Komponente, die man in die Sammlung eingefügt hat, steht an einer bestimmten Position, die durch einen Index bezeichnet wird. Ein Index ist ein nicht-negativer `int`-Wert.

5.6. Durch welche Methode in welcher Schnittstelle wird die *natürliche Ordnung* einer Klasse festgelegt?

Durch die Methode `compareTo` der Schnittstelle `Comparable<T>`.

Lösung 6 (15 Punkte) : Zeitkomplexitäten

zk01: $O(n)$

zk02: $O(n^3)$

zk03: $O(n^2)$

zk04: $O(n^5)$

zk05: $O(n^3)$

zk06: $O(n)$