

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): Ein *Palindrom* ist eine Zeichenkette, die gleich bleibt, wenn man die Reihenfolge ihrer Zeichen umkehrt. Beispiele für Palindrome:

anna, rentner, lagerregal, dieliebeistsiegerstetsregeistsiebeileid.

Mit der folgenden Methode kann man feststellen, ob ein String *s* ein Palindorm ist oder nicht:

```
1  static public boolean istPalindrom(String s) {
2      // Liefert true genau dann wenn s ein Palindrom ist.
3      if (s.length() <= 1) return true;
4      return istPalindromR(s, 0, s.length()-1);
5  } // istPalindrom
```

Allerdings erledigt diese Methode *istPalindrom* nur triviale Sonderfälle und überläßt die meiste Arbeit einer weiteren Methode. Diese Methode namens *istPalindromR* sollen Sie als **rekursive Methode** programmieren:

```
6  static private boolean istPalindromR(String s, int von, int bis) {
7      // Diese Funktion verlaesst sich darauf, dass sie nur von der
8      // Funktion istPalindrom aufgerufen wird. Sie hilft dabei, die
9      // Beschreibung in Zeile 2 zur erfuellen.
10
11     // In istPalindromR sollen keine neuen Objekte erzeugt werden
12     // (weil das unnoetig teuer ist), die String-Methode substring
13     // soll nicht aufgerufen werden (weil sie ein neues String-Objekt
14     // erzeugt) und istPalindromR soll keine Schleifen enthalten
15     // (sondern "rekursiv funktionieren").
16     ...
17 } // istPalindromR
```

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

```
1  static public int wertVonIntX(Object ob) {
2      // Falls ob ein oeffentliches int-Attribut namens x enthaelt,
3      // wird der Wert dieses Attributs geliefert. Sonst wird
4      // Integer.MIN_VALUE geliefert.
5      ...
6  }
```

Aufgabe 3 (15 Punkte): Stellen Sie die folgenden Variablen als Bojen dar und beantworten Sie dann die darunter stehenden Fragen:

```
1  String      stA = "Hallo!";
2  String      stB = "Hallo!";
3  String      stC = new String("Hallo!");
4  StringBuilder sbD = new StringBuilder("Hallo!");
5  StringBuilder sbE = new StringBuilder("Hallo!");
```

Welche Werte haben die folgenden Ausdrücke:

- 3.1. stA == stB
- 3.2. stA == stC
- 3.3. stA.equals(stB)
- 3.4. stA.equals(stC)
- 3.5. sbD == sbE
- 3.6. sbD.equals(sbE)

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

```

1 <?xml version="1.0" encoding="UTF-8"?>
2 <!-- -----
3     Datei kfz.dtd:
4 ----- -->
5 <!ELEMENT kfz      (bezeich?, halter, kennzeichen?)>
6 <!ELEMENT bezeich  (#PCDATA)>
7 <!ELEMENT halter   (vorName+, nachName, adresse)>
8 <!ELEMENT kennzeichen (#PCDATA)>
9 <!ELEMENT vorName  (#PCDATA)>
10 <!ELEMENT nachName (#PCDATA)>
11 <!ELEMENT adresse  (#PCDATA)>
12
13 <!ATTLIST kfz
14   art      (pkw|lkw|krad|sonst) "pkw"
15 >
16 <!ATTLIST kennzeichen
17   kuerzel  (B|HVL|OHV|P)  #REQUIRED
18   ort      CDATA           #IMPLIED
19 >

```

Geben Sie für jedes der folgenden sechs XML-Dokumente an, ob es bezüglich dieser DTD *gültig* oder *nicht gültig* ist. Beschreiben Sie ausserdem für jedes nicht-gültige Dokument kurz, *warum* es nicht gültig ist.

<pre> 1 Dokument KfzA.xml 2 <kfz art="pkw"> 3 <bezeich>Polo Joker</bezeich> 4 <halter> 5 <vorName>Joshua</vorName> 6 <nachName>Bloch</nachName> 7 <adresse>Kudamm 17</adresse> 8 </halter> 9 <kennzeichen kuerzel="B" 10 ort="Berlin"> 11 12345 12 </kennzeichen> 13 </kfz> </pre>	<pre> Dokument KfzB.xml <kfz art="motorrad"> <halter> <vorName>Karl</vorName> <vorName>Maria</vorName> <nachName>Brandauer</nachName> <adresse>Kudamm 17</adresse> </halter> <kennzeichen kuerzel="HVL"> 12345 </kennzeichen> </kfz> </pre>
<pre> 14 Dokument KfzC.xml 15 <kfz art="sonst"> 16 <bezeich>Kudamm 17 17 </bezeich> 18 <halter> 19 <vorName>Maier</vorName> 20 <nachName>Karl</nachName> 21 <adresse>Citroen DS</adresse> 22 </halter> 23 <kennzeichen kuerzel="OHV" 24 ort="Unterhavel"> 25 ABCDEF 26 </kennzeichen> 27 </kfz> </pre>	<pre> Dokument KfzD.xml <kfz art="pkw"> <halter> <vorName>James</vorName> <nachName>Fenimor</nachName> <nachName>Cooper</nachName> <adresse>Schsendamm 13 </adresse> </halter> </kfz> </pre>
<pre> 28 Dokument KfzE.xml 29 <kfz> 30 <halter> 31 <vorName>James</vorName> 32 <vorName>Prescott</vorName> 33 <nachName>Joule</nachName> 34 <adresse>Mercedes 500</adresse> 35 </halter> 36 <kennzeichen ort="Berlin"> 37 12345 38 </kennzeichen> 39 </kfz> 40 </pre>	<pre> Dokument KfzF.xml <kfz art="lkw"> <bezeich>Mercedes 220</bezeich> <halter> <vorName>Joshua</vorName> <nachName>Bloch</nachName> <adresse>Kudamm 17</adresse> </halter> <kennzeichen kuerzel="B" ort="Potsdam"> 12345 </kennzeichen> </kfz> </pre>

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. Geben Sie die Namen von (mind.) zwei *Listener-Schnittstellen* an.
- 5.2. Welche Java-Klassen gelten als *Sammlungsklassen* (engl. collection classes)?
- 5.3. Wann ist ein binärer Baum *sortiert*?
- 5.4. Welche *Tiefe* hat ein binärer Baum mit 4 Millionen Knoten mindestens?
- 5.5. Skizzieren Sie (beschreiben Sie ganz kurz) eine Situation, in der man eine *Auszeichnungssprache* typischerweise benutzt.
- 5.6. Was muss für alle Objekte k_1 und k_2 einer Comparable<K>-Klasse K gelten, damit man sagen kann: Die equals-Methode und die compareTo-Methode in einem K-Objekt sind *konsistent* (zueinander)?

Aufgabe 6 (15 Punkte): Geben Sie für jede der Methoden zk01 bis zk05 an, welche *Zeitkomplexität* sie hat (z.B. $O(1)$ oder $O(n)$ 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<500; i++) {
3          machWas();
4      }
5  } // zk01
6
7  static void zk02(int n) {
8      for (int i=0; i<n*n*n; i++) {
9          machWas();
10     }
11 } // zk02
12
13 static void zk03(int n) {
14     for (int i=0; i<3*n; i++) {
15         for (int j=0; j<2*n; j++) {
16             machWas();
17         }
18     }
19 } // zk03
20
21 static void zk04(int n) {
22     if (n<=0) {
23         return;
24     } else {
25         machWas();
26         zk04(n-1);
27     }
28 } // zk04
29
30 static void zk05(int n) {
31     if (n<=1) {
32         machWas();
33     } else {
34         machWas();
35         zk05(n-1);
36         zk05(n-1);
37     }
38 } // zk05

```

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 boolean istPalindrom(String s) {
2      // Liefert true genau dann wenn s ein Palindrom ist.
3      if (s.length() <= 1) return true;
4      return istPalindrom2(s, 0, s.length()-1);
5  } // istPalindrom
6
7  // Die folgende Methode soll programmiert werden
8  static private boolean istPalindrom2(String s, int von, int bis) {
9      // Diese Funktion verlaesst sich darauf, dass sie nur von der
10     // Funktion istPalindrom aufgerufen wird.
11
12     if (bis-von <= 0) return true;
13     if (s.charAt(von) != s.charAt(bis)) return false;
14     return istPalindrom2(s, von+1, bis-1);
15 } // istPalindrom2

```

Lösung 2 (20 Punkte):

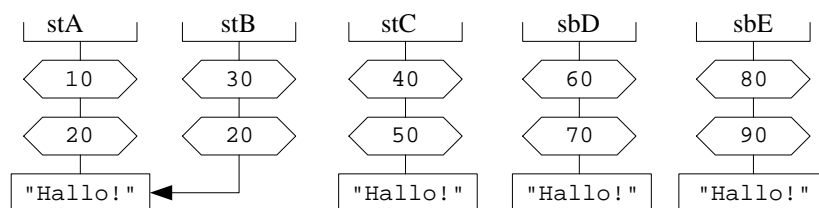
```

1  static public int wertVonIntX(Object ob) {
2      // Falls ob ein oeffentliches int-Attribut namens x enthaelt,
3      // wird der Wert dieses Attributs geliefert. Sonst wird
4      // Integer.MIN_VALUE geliefert.
5
6      try {
7          Class<?> kob = ob.getClass();
8          Field[] fr    = kob.getFields();
9
10         for (Field f : fr) {
11             if (!"x".equals(f.getName())) continue;
12             if (f.getType() != Integer.TYPE) continue;
13             return f.getInt(ob);
14         }
15     } catch (IllegalAccessException ex) {}
16     return Integer.MIN_VALUE;
17 } // wertVonIntX

```

Lösung 3 (15 Punkte):

Wie sehen die Variablen stA bis sdD (als *Bojen* dargestellt) aus:

**Ausdrücke und ihre Werte (mit Erläuterungen):**

```

stA == stB:      true   (weil 20 gleich 20 ist)
stA == stC:      false  (weil 20 ungleich 50 ist)
stA.equals(stB): true   (weil "Hallo!" gleich "Hallo!" ist)
stA.equals(stC): true   (weil "Hallo!" gleich "Hallo!" ist)
sbD == sbE:      false  (weil 70 ungleich 90 ist)
sbD.equals(sbE): false  (weil 70 ungleich 90 ist)

```

Lösung 4 (15 Punkte):

```
1 istGueltig(KfzA.xml): true
2 DOMError (error): Attribute "art" with value "motorrad" must
3 have a value from the list "pkw lkw krad sonst ".
4 istGueltig(KfzB.xml): false
5 istGueltig(KfzC.xml): true
6 DOMError (error): The content of element type "halter" must
7 match "(vorName+,nachName,adresse)".
8 istGueltig(KfzD.xml): false
9 DOMError (error): Attribute "kuerzel" is required and must
10 be specified for element type "kennzeichen".
11 istGueltig(KfzE.xml): false
12 istGueltig(KfzF.xml): true
```

Lösung 5 (15 Punkte):

5.1. 4. Geben Sie die Namen von (mind.) zwei Listener-Schnittstellen an.

MouseListener, MouseMotionListener, MouseWheelListener, WindowListener, ...

5.2. Welche Java-Klassen gelten als *Sammlungsklassen* (engl. collection classes)?

Alle Klassen, die die Schnittstelle Collection implementieren.

5.3. Wann ist ein binärer Baum sortiert?

Wenn für jeden Knoten k des Baumes gilt: Der Schlüssel von k ist größer als alle Schlüssel im linken Unterbaum von k und kleiner als alle Schlüssel im rechten Unterbaum von k .

5.4. Welche Tiefe hat ein binärer Baum mit 4 Millionen Knoten mindestens?

Er hat mindestens die Tiefe 22.

5.5. Skizzieren Sie (beschreiben Sie ganz kurz) eine Situation, in der man eine *Auszeichnungssprache* typischerweise benutzt.

Wenn ein Programm Daten ausgibt, die von einem anderen Programm weiterverarbeitet werden sollen.

5.6. Was muss für alle Objekte k_1 und k_2 einer Comparable<K>-Klasse K gelten, damit man sagen kann: Die equals-Methode und die compareTo-Methode in einem K -Objekt sind *konsistent* (zueinander)?

Der Ausdruck equals(k_1 , k_2) muss genau dann den Wert true haben, wenn der Ausdruck $k_1.compareTo(k_2)$ den Wert 0 hat.

Lösung 6 (15 Punkte) : Zeitkomplexitäten

zk01: $O(1)$

zk02: $O(n^3)$

zk03: $O(n^2)$

zk04: $O(n)$

zk05: $O(2^n)$