

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 (25 Punkte): Zwei voneinander unabhängige Teilaufgaben.

Beide müssen **rekursiv** gelöst werden (sie dürfen keine Schleifen enthalten).

```

1  // -----
2  // Teilaufgabe 1.1 (15 Punkte): Die Prozedur gibAus ist fertig vorgegeben.
3  // Sie sollen nur die Prozedur gibAusR (als rekursive Methode) schreiben.
4  // Die beiden Methoden dürfen ihren Reihungsparameter b nicht verändern.
5
6  static public void gibAus(int[] b) {
7      // Verlaesst sich darauf, dass b ein (aufsteigend) sortierter Baum
8      // von positiven Zahlen ist und dass alle leeren Komponenten den
9      // Wert -1 haben.
10     // Gibt die Zahlen (aus den nicht-leeren Komponenten) in absteigender
11     // Reihenfolge zur Standardausgabe aus (alle auf einer Zeile, mit
12     // einem Blank " " hinter jeder Zahl).
13     // Zur Erinnerung an Baeume in Reihungen:
14     // b[0] ist unbenutzt, b[1] ist die Wurzel des Baumes.
15     // Bei b[2*i] beginnt der linke Unterbaum von b[i].
16     // Bei b[2*i+1] beginnt der rechte Unterbaum von b[i].
17     //
18     // Beispiel fuer die Wirkung von gibAus:
19     // int[] b1 = {0, 17, 12, 26, -1, 15, -1, -1};
20     // Der Prozeduraufruf gibAus(b1) gibt dann folgende Zeile aus:
21     // 26 17 15 12
22
23     gibAusR(b, 1);
24     pln();
25 } // gibAus
26
27 static private void gibAusR(int[] b, int hier) {
28     // Hilfsmethode fuer gibAus.
29     // Soll von Ihnen programmiert werden.
30     ...
31 } // gibAusR
32
33 // -----
34 // Teilaufgabe 1.2 (10 Punkte): Eine rekursive Funktion
35
36 static public String als7erZahl(long n) {
37     // Liefert n als 7-er-Zahl (als Zahl im Zahlensystem mit der Basis 7),
38     // negative Zahlen mit einem Minuszeichen - davor.
39     // Beispiele:
40     // als7erZahl(15) ist gleich "21"
41     // als7erZahl(-9) ist gleich "-12"
42     // als7erZahl(49) ist gleich "100"
43     // als7erZahl(0) ist gleich "0"
44     //
45     // Wichtige Anforderung: Auch diese Methode muss rekursiv
46     // programmiert werden (sie darf keine Schleifen enthalten).
47     ...
48 } // als7erZahl

```

Aufgabe 2 (15 Punkte): Schreiben Sie eine Funktion, die der folgenden Spezifikation entspricht (dabei dürfen Sie alle nützlichen `import`-Befehle *voraussetzen* und brauchen Sie *nicht* selbst hinzuschreiben):

```

1  static public Document createJdomDocument() {
2      // Liefert ein Objekt des Typs org.jdom.Document, welches
3      // das folgende XML-Dokument repraesentiert:
4      //
5      // <?xml version="1.0" encoding="UTF-8"?>
6      // <zitat>
7      //   <autor>Murphy<\autor>
8      //   <inhalt sprache="Englisch">
9      //     If anything can go wrong, it will.
10     //   <\inhalt>
11     // <\zitat>
12     // ...
13   } // createJdomDocument

```

Aufgabe 3 (15 Punkte): Beantworten Sie die folgenden Fragen möglichst kurz, aber genau.

1. Welche (grundlegende, charakterisierende) Eigenschaft haben sowohl *Prozesse* eines Betriebssystems als auch *Fäden* (engl. threads) eines Programms?
2. Skizzieren Sie (ganz kurz) ein Problem, zu dessen Lösung man *reproduzierbare Zufallszahlen* verwendet.
3. Wenn man Algorithmen analysiert um ihre Zeitkomplexität herauszufinden, was für Befehlsfolgen darf man dabei als *einen Schritt* betrachten?
4. *Wie viele Schritte* braucht man höchstens, um in einer sortierten Reihung der Länge 16 Millionen in einem positiven Fall zu suchen? Geben Sie die konkrete ("ausgerechnete") Zahl an, keine Formel.
5. Was ist eine *Hash-Tabelle*? Geben Sie möglichst die im SU behandelte Kurzdefinition an.
6. Welche "*weiche Bedingung*" müssen Set-Sammlungen (z.B. HashSet, TreeSet etc.) einhalten?
7. Wie heißen die beiden wichtigsten Sprachen, in denen man *Typen* von XML-Dateien beschreiben kann (z.B. Dateitypen wie Rechnung, Formel, Internetseite etc.)?
8. In der Klasse `java.lang.Class<T>` wird eine Methode `getMethods()` mit 0 Parametern vereinbart. Welchen *Rückgabety*p hat diese Funktion? *Was* (für Informationen) liefert diese Funktion?

Aufgabe 4 (15 Punkte): Schreiben Sie eine Funktion, die der folgenden Spezifikation entspricht:

```

1  static public <K> Collection<K> macheSammlung(K... kr) {
2      // Liefert eine Sammlung, die alle Parameter als Komponenten
3      // enthaelt.
4      // Beispiele:
5      // pln(macheSammlung(10, 20, 30, 40, 50)); gibt aus: [10, 20, 30, 40, 50]
6      // pln(macheSammlung("ABC", "DE", FGHI)); gibt aus: [ABC, DE, FGHI]
7      // pln(macheSammlung());                gibt aus: []
8      // ...
9  } // macheSammlung

```


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 (25 Punkte): Zwei voneinander unabhängige Teilaufgaben. Beide müssen rekursiv gelöst werden (sie dürfen keine Schleifen enthalten).

```

26 // -----
27 // Teilaufgabe 1.1 (15 Punkte): Die Prozedur gibAus ist fest vorgegeben.
28 // Sie sollen nur die Prozedur gibAusR (als rekursive Methode) schreiben.
1  // Die beiden Methoden dürfen ihren Reihungsparameter b nicht verändern.
2  static public void gibAus(int[] b) {
3      // Verlaesst sich darauf, dass b ein (aufsteigend) sortierter Baum
4      // von positiven Zahlen ist und dass alle leeren Komponenten den
5      // Wert -1 haben.
6      // Gibt die ("nicht-leeren") Zahlen in absteigender Reihenfolge
7      // zur Standardausgabe aus (alle auf einer Zeile, mit einem
8      // Blank " " hinter jeder Zahl).
9      // Zur Erinnerung an Baeume in Reihungen:
10     // b[0] ist unbenutzt, b[1] ist die Wurzel des Baumes.
11     // Bei b[2*i] beginnt der linke Unterbaum von b[i].
12     // Bei b[2*i+1] beginnt der rechte Unterbaum von b[i].
13     //
14     // Beispiel fuer die Wirkung von gibAus:
15     // int[] b1 = {0, 17, 12, 26, -1, 15, -1, -1};
16     // Der Prozeduraufruf gibAus(b1) gibt dann folgende Zeile aus:
17     // 26 17 15 12
18
19     gibAusR(b, 1);
20     pln();
21 } // gibAus
22
23 static private void gibAusR(int[] b, int hier) {
24     // Hilfsmethode fuer gibAus.
25     if (hier >= b.length) return;
26     if (b[hier] == -1) return;
27     gibAusR(b, 2*hier+1);
28     p(b[hier] + " ");
29     gibAusR(b, 2*hier);
30 } // gibAusR

1  // -----
1  // Teilaufgabe 1.2 (10 Punkte): Eine rekursive Funktion
2
3  static public String als7erZahl(long n) {
4      // Liefert n als 7-er-Zahl (als Zahl im Zahlensystem mit der Basis 7),
5      // negative Zahlen mit einem Minuszeichen - davor.
6      // Beispiele:
7      // als7erZahl(15) ist gleich "21"
8      // als7erZahl(-9) ist gleich "-12"
9      // als7erZahl(49) ist gleich "100"
10     // als7erZahl(0) ist gleich "0"
11     //
12     // Wichtige Anforderung: Auch diese Methode muss rekursiv
13     // programmiert werden (sie darf keine Schleifen enthalten).
14
15     if (n < 0) return "-" + als7erZahl(-n);
16     if (n < 7) return "" + n;
17
18     return als7erZahl(n/7) + n%7;
19
20 } // als7erZahl

```

Lösung 2 (15 Punkte):

```

1  static public Document createJdomDocument() {
2      // Liefert ein Objekt des Typs org.jdom.Document, welches
3      // das folgende XML-Dokument repräsentiert:
4      //
5      // <?xml version="1.0" encoding="UTF-8"?>
6      // <zitat>
7      //   <autor>Murphy<\autor>
8      //   <inhalt sprache="Englisch">
9      //     If anything can go wrong, it will.
10     //   <\inhalt>
11     // <\zitat>
12
13     Element eZitat = new Element("zitat");
14     Element eAutor = new Element("autor");
15     Element eInhalt = new Element("inhalt");
16
17     eInhalt.setAttribute("sprache", "Englisch");
18     eZitat.setContent(eAutor);
19     eZitat.addContent(eInhalt);
20
21     eAutor.setText("Murphy");
22     eInhalt.setText("If anything can go wrong, it will.");
23
24     Document erg = new Document(eZitat);
25     return erg;
26 } // createJdomDocument

```

Lösung 3 (15 Punkte) : Fragen

1. Welche (grundlegende, charakterisierende) Eigenschaft haben sowohl *Prozesse* eines Betriebssystems als auch *Fäden* (engl. threads) eines Programms?

Sowohl Prozesse als auch Fäden werden nebenläufig zueinander (concurrently) ausgeführt.

2. Skizzieren Sie (ganz kurz) ein Problem, zu dessen Lösung man *reproduzierbare Zufallszahlen* verwendet.

Erstellen von (großen Mengen von) Testdaten.

3. Wenn man Algorithmen analysiert um ihre Zeitkomplexität herauszufinden, was für Befehlsfolgen darf man dabei als *einen Schritt* betrachten?

Jede Befehlsfolge von der es plausibel ist anzunehmen, dass ein Ausführer sie in einer bestimmten Zeit (die unabhängig von der Problemgröße ist) ausführen kann.

4. Wie viele Schritte braucht man höchstens, um in einer sortierten Reihung der Länge 16 Millionen in einem positiven Fall zu suchen? Geben Sie die konkrete ("ausgerechnete") Zahl an, keine Formel.
24 Schritte.

5. Was ist eine *Hash-Tabelle*? Geben Sie möglichst die im SU behandelte Kurzdefinition an.
Eine Reihung von Listen.

6. Welche "*weiche Bedingung*" müssen Set-Sammlungen (z.B. HashSet, TreeSet etc.) einhalten?
Set-Sammlungen dürfen keine Doppelgänger enthalten.

7. Wie heißen die beiden wichtigsten Sprachen, in denen man *Typen* von XML-Dateien beschreiben kann (z.B. Dateitypen wie Rechnung, Formel, Internetseite etc.)?
DTD (Document Type Definition) und XML Schema.

8. In der Klasse `java.lang.Class<T>` wird eine Methode `getMethods()` mit 0 Parametern vereinbart. Welchen Rückgabotyp hat diese Funktion? Was (für Informationen) liefert diese Funktion?

Rückgabotyp: Reihung von Method. Die Ergebnisreihung enthält für jede (vereinbarte oder geerbte) öffentliche Methode der betreffenden Klasse ein Method-Objekt.

Lösung 4 (20 Punkte): Schreiben Sie eine Funktion, die der folgenden Spezifikation entspricht:

```

1  static public <K> Collection<K> macheSammlung(K... kr) {
2      // Liefert eine Sammlung, die alle Parameter als Komponenten
3      // enthaelt.
4      // Beispiele:
5      // pln(macheSammlung(10, 20, 30, 40, 50)); gibt aus: [10, 20, 30, 40, 50]
6      // pln(macheSammlung("ABC", "DE", FGHI)); gibt aus: [ABC, DE, FGHI]
7      // pln(macheSammlung());                       gibt aus: []
8
9      Collection<K> alk = new ArrayList<K>();
10     for (K k : kr) alk.add(k);
11     return alk;
12 } // macheSammlung

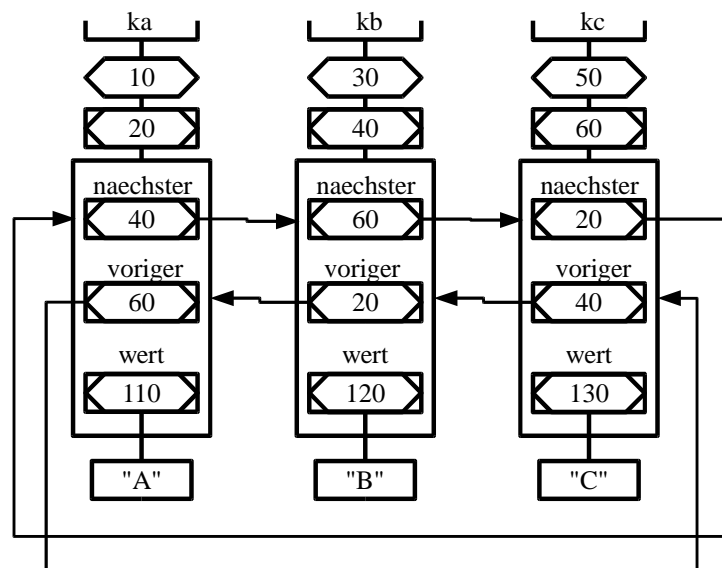
```

Lösung 5 (15 Punkte):

Angenommen, der Ausführer hat die Methode aufgabe5 bis Zeile 24 ausgeführt.

Wie sehen die Variablen ka, kb und kc in diesem Moment aus?

Stellen Sie die Variablen als Bojen dar (in vereinfachter oder in ausführlicher Darstellung).



Lösung 6 (15 Punkte):

printf-Befehl:	Ausgabe des printf-Befehles											
<code>printf("A %s%s %1\$s%2\$s %1\$s%2\$s A%n", "X", "Y");</code>	A		X	Y		X	Y		X	Y		A
<code>printf("B %s%s %1\$s%<s %2\$s%<s B%n", "X", "Y");</code>	B		X	Y		X	X		Y	Y		B
<code>printf("C %s%3\$s %s%<s %s%1\$s C%n", "X", "Y", "Z");</code>	C		X	Z		Y	Y		Z	X		C
<code>printf("D %+ ,8.2f D%n", -34.567);</code>	D				-	3	4	,	5	7		D
<code>printf("E %- ,8.2f E%n", 34.567);</code>	E		3	4	,	5	7					E
<code>printf("F %0 ,8.2f F%n", -34.567);</code>	F		-	0	0	3	4	,	5	7		F
<code>printf("G %+ ,8d G%n", 34567);</code>	G			+	3	4	.	5	6	7		G
<code>printf("H %- ,8d H%n", -34567);</code>	H		-	3	4	.	5	6	7			H

Wie es auf dem Bildschirm aussieht:

A XY XY XY **A**
B XY XX YY **B**
C XZ YY ZX **C**
D -34,57 **D**
E 34,57 **E**
F -0034,57 **F**
G +34.567 **G**
H -34.567 **H**