

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): Schreiben Sie zwei Methoden, die den folgenden Spezifikationen entsprechen. Die Funktionen ähneln sich ein bisschen, sind aber **unabhängig** voneinander.

Wichtige Anforderung: Beide Funktionen müssen **rekursiv** arbeiten, nicht mit Schleifen!

```

1  static public int anzRobosA(int tage) {
2      // Wenn wir 3 Roboter haben, die (zusammen) an einem Tag 1 weiteren
3      // Roboter bauen koennen, und wir tage viele Tage Roboter bauen lassen
4      // (von allen Robotern, die wir dann schon haben), wie viele Roboter
5      // haben wird dann am Ende? Diese Funktion liefert die Antwort.
6      // Beispiele:
7      //
8      // anzRobosA(0) ist 3   (zu wenig Zeit um Robos zu bauen)
9      // anzRobosA(1) ist 4   (es wurde 1 Robo gebaut)
10     // anzRobosA(2) ist 5   (es wurden 1+1 Robos gebaut)
11     // anzRobosA(3) ist 6   (es wurden 1+1+1 Robos gebaut)
12     // anzRobosA(4) ist 8   (es wurden 1+1+1+2 Robos gebaut)
13     // anzRobosA(5) ist 10  (es wurden 1+1+1+2+2 Robos gebaut)
14     // anzRobosA(6) ist 13  (es wurden 1+1+1+2+2+3 Robos gebaut)
15     ...
16 } // anzRobosA

```

```

1  static public int anzRobosB(int habSchon, int proArbGruppe, int tage) {
2      // Hier werden Roboter nach folgenden Regeln gebaut:
3      // Eine Arbeitsgruppe besteht aus proArbGruppe vielen Robotern.
4      // Eine Arbeitsgruppe kann pro Tag einen weiteren Roboter bauen.
5      // Am Anfang haben wir habSchon viele Roboter.
6      // Wir lassen tage viele Tage Roboter bauen (so viele wie moeglich).
7      // Wie viele Roboter haben wir dann am Ende?
8      // Diese Funktion liefert die Antwort. Beispiele:
9      //
10     // anzRobosB(5, 2, 0) ist 5   (zu wenig Zeit um Robos zu bauen)
11     // anzRobosB(2, 3, 10) ist 2   (zu wenig Robos um Robos zu bauen)
12     // anzRobosB(3, 3, 1) ist 4   (es wurde 1 Robo gebaut)
13     // anzRobosB(3, 3, 3) ist 6   (es wurden 1+1+1 Robos gebaut)
14     // anzRobosB(3, 3, 4) ist 8   (es wurden 1+1+1+2 Robos gebaut)
15     // anzRobosB(5, 3, 5) ist 17  (es wurden 1+2+2+3+4 Robos gebaut)
16     // anzRobosB(5, 1, 3) ist 40  (es wurden 5+10+20 Robos gebaut)
17     ...
18 } // anzRobosB

```

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      // <rechnung nr="12345">
7      //   <posten>33,33</posten>
8      //   <posten>12,34</posten>
9      // </rechnung>
10     ...
11 } // createJdomDocument

```

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

1. Geben Sie einen (einfachen aber präzise beschreibbaren) Unterschied zwischen dem *parametrisierten Typ* `ArrayList<Object>` und dem *rohen Typ* `ArrayList` an.

2. Angenommen, `f01` ist ein Field-Objekt. Was bewirkt dann der Methodenaufruf `f01.setAccessible(true);`?

3. Die Schnittstelle `Comparator<T>` enthält nur *eine* Methode. Geben Sie ihr *Profil* an.

4. Die folgende String-Variable besteht aus 4 Teilen:

```
String st1 = new String("Hallo");
```

Welchen dieser Teile bezeichnet der Name `st1` in den folgenden Zusammenhängen:

4.1. ... `st1 == "Hallo"` ...

4.2. ... `st1.equals("Hallo")` ...

4.3. ... `st1 = "Hello";` ...

4.4. ... `println(st1);` ...

5. Binäre Bäume, ihre Tiefe und die Anzahl ihrer Knoten:

5.1. *Wie viele Knoten* kann ein binärer Baum der Tiefe 23 höchstens haben (ungefähr)?

5.2. *Welche Tiefe* hat ein binärer Baum mit 10 Tausend Knoten mindestens (genau)?

6. Was ist ein Modul?

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

```
1  static public int anzVokale(ArrayList<String[]> alsr) {
2      // Wie viele gross Vokale ('A', 'E', 'I', 'O', 'U')
3      // kommen in (den Komponenten der Komponenten von) alsr vor?
4      // Diese Funktion liefert die Antwort. Zwei Beispiele:
5      //
6      // String[] sr0 = {"ALAMO", "aaa", "UUU"};           // 6 grosse Vokale
7      // String[] sr1 = {"A", "E", "I", "O", "U", ""};      // 5 grosse Vokale
8      // String[] sr2 = {};                                // 0 grosse Vokale
9      // ArrayList<String[]> alsr01 = new ArrayList<String[]>();
10     // ArrayList<String[]> alsr02 = new ArrayList<String[]>();
11     // alsr01.add(sr0);
12     // alsr01.add(sr1);
13     // alsr01.add(sr2);
14     // anzVokale(alsr01) ist gleich 11
15     // anzVokale(alsr02) ist gleich 0
16     ...
17 }
```

Aufgabe 5 (15 Punkte): Betrachten Sie die folgende Klasse `Knoten` und die Methode `aufgabe5` (die zu irgendeiner anderen Klasse gehört, die hier nicht dargestellt wird):

```

1  class Knoten {
2      Knoten next;
3      int wert;
4
5      Knoten(Knoten next, int wert) {
6          this.next = next;
7          this.wert = wert;
8      }
9  }
10
11  static public void aufgabe5() {
12      Knoten k4 = new Knoten(null, 4);
13      Knoten k3 = new Knoten(k4, 3);
14      Knoten k2 = new Knoten(k4, 2);
15      Knoten k1 = new Knoten(k3, 1);
16      ...
17      k4.next = k1.next;
18      k3.next = k1;
19      k2.next = k3.next;
20      k1.next = null;
21      ...
22  } // aufgabe5

```

Angenommen, der Ausführer führt die Methode `aufgabe5` aus.

Wie sehen die Variablen `k1` bis `k4` aus (als vereinfachte oder ausführliche Bojen dargestellt), wenn er

1. die Zeile 15 erreicht?
2. die Zeile 20 erreicht?

Aufgabe 6: (15 Punkte)

Was geben die folgenden `printf`-Befehle zum Bildschirm aus? Notieren Sie Ihre Lösungen so sauber und lesbar, dass klar zu erkennen ist, wie viele und welche Zeichen ausgegeben werden. Sie können Ihre Lösungen wahlweise direkt neben der Aufgabe in die vorgegebenen Kästchen eintragen (*ein* Zeichen pro Kästchen) oder auf ein separates Blatt schreiben:

printf(" A %+,8d \n", 12345);	+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+
printf(" B %+,8d \n", -12345);	+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+
printf(" C %- ,8d \n", 12345);	+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+
printf(" D %- ,8d \n", -12345);	+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+
printf(" E %0,8d \n", 12345);	+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+
printf(" F %0,8d \n", -12345);	+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+
printf(" G %3.5s \n", 12345);	+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+
printf(" H %3.5s \n", -12345);	+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+

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):

```

1  static public int anzRobosA(int tage) {
2      // Wenn wir 3 Roboter haben, die (zusammen) an einem Tag 1 weiteren
3      // Roboter bauen koennen, und wir tage viele Tage Roboter bauen lassen
4      // (von allen Robotern, die wir dann schon haben), wie viele Roboter
5      // haben wird dann am Ende? Diese Funktion liefert die Antwort.
6      // Beispiele:
7      //
8      // anzRobosA(0) ist 3 (zu wenig Zeit um Robos zu bauen)
9      // anzRobosA(1) ist 4 (es wurde 1 Robo gebaut)
10     // anzRobosA(2) ist 5 (es wurden 1+1 Robos gebaut)
11     // anzRobosA(3) ist 6 (es wurden 1+1+1 Robos gebaut)
12     // anzRobosA(4) ist 8 (es wurden 1+1+1+2 Robos gebaut)
13     // anzRobosA(5) ist 10 (es wurden 1+1+1+2+2 Robos gebaut)
14     // anzRobosA(6) ist 13 (es wurden 1+1+1+2+2+3 Robos gebaut)
15
16     if (tage == 0) return 3;
17
18     int hatteVorDemLetztenTag = anzRobosA(tage-1);
19     int habeAmLetztenTagGebaut = hatteVorDemLetztenTag / 3;
20     return hatteVorDemLetztenTag + habeAmLetztenTagGebaut;
21 } // anzRobosA

1  static public int anzRobosB(int habSchon, int proArbGruppe, int tage) {
2      // Hier werden Roboter nach folgenden Regeln gebaut:
3      // Eine Arbeitsgruppe besteht aus proArbGruppe vielen Robotern.
4      // Eine Arbeitsgruppe kann pro Tag einen weiteren Roboter bauen.
5      // Am Anfang haben wir habSchon viele Roboter.
6      // Wir lassen tage viele Tage Roboter bauen (so viele wie moeglich).
7      // Wie viele Roboter haben wir dann am Ende?
8      // Diese Funktion liefert die Antwort. Beispiele:
9      //
10     // anzRobosB(5, 2, 0) ist 5 (zu wenig Zeit um Robos zu bauen)
11     // anzRobosB(2, 3, 10) ist 2 (zu wenig Robos um Robos zu bauen)
12     // anzRobosB(3, 3, 1) ist 4 (es wurde 1 Robo gebaut)
13     // anzRobosB(3, 3, 3) ist 6 (es wurden 1+1+1 Robos gebaut)
14     // anzRobosB(3, 3, 4) ist 8 (es wurden 1+1+1+2 Robos gebaut)
15     // anzRobosB(5, 3, 5) ist 17 (es wurden 1+2+2+3+4 Robos gebaut)
16     // anzRobosB(5, 1, 3) ist 40 (es wurden 5+10+20 Robos gebaut)
17
18     if (habSchon < proArbGruppe || tage == 0) return habSchon;
19
20     // Loesung analog zu anzRobosA:
21     int anzVorDemLetztenTag = anzRobosB(habSchon, proArbGruppe, tage-1);
22     int anzAmLetztenTagGebaut = anzVorDemLetztenTag / proArbGruppe;
23     return anzVorDemLetztenTag + anzAmLetztenTagGebaut;
24
25     // Eine alterntive Loesung:
26     // int habSchonNachEinemTag = habSchon + habSchon/proArbGruppe;
27     // return anzRobosB(habSchonNachEinemTag, proArbGruppe, tage-1);
28 } // anzRobosB

```

Lösung 2 (15 Punkte):

```

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      // <rechnung nr="12345">
7      //   <posten>33,33</posten>
8      //   <posten>12,34</posten>
9      // </rechnung>
10
11     Element eRechnung = new Element("rechnung");
12     Element ePosten1  = new Element("posten");
13     Element ePosten2  = new Element("posten");
14
15     eRechnung.setAttribute("nr", "12345");
16     eRechnung.setContent(ePosten1);
17     eRechnung.addContent(ePosten2);
18
19     ePosten1.setText("33,33");
20     ePosten2.setText("12,34");
21
22     Document erg = new Document(eRechnung);
23     return erg;
24 } // createJdomDocument

```

Lösung 3 (15 Punkte) : Fragen

1. Was ist der Unterschied zwischen dem *parametrisierten Typ* `ArrayList<Object>` und dem *rohen Typ* `ArrayList`?

Der rohe Typ `ArrayList` ist ein Obertyp aller parametrisierten Typen wie `ArrayList<Integer>`, `ArrayList<JTextField>`, `ArrayList<Object>`, Der parametrisierte Typ `ArrayList<Object>` ist weder Ober- noch Untertyp anderer parametrisierten Typen wie `ArrayList<Integer>` oder `ArrayList<JTextField>` etc.

2. Angenommen, `f01` ist ein Field-Objekt. Was bewirkt dann der Methodenaufruf `f01.setAccessible(true)`?

Er bewirkt, dass man auf den Wert des durch `f01` reflektierten Attributes zugreifen darf, auch wenn es nicht öffentlich (public) ist.

3. Die Schnittstelle `Comparator<T>` enthält nur *eine* Methode. Geben Sie ihr Profil an.

int compare T T

4. Die folgende String-Variable besteht aus 4 Teilen:

```
String st1 = new String("Hallo");
```

Welchen dieser Teile bezeichnet der Name `st1` in den folgenden Zusammenhängen:

- | | |
|---|---|
| 4.1. ... <code>st1 == "Hallo"</code> ... | <code>st1</code> bezeichnet den Wert |
| 4.2. ... <code>st1.equals("Hallo")</code> ... | <code>st1</code> bezeichnet den Zielwert |
| 4.3. ... <code>st1 = "Hello";</code> ... | <code>st1</code> bezeichnet den Wert |
| 4.4. ... <code>pln(st1);</code> ... | <code>st1</code> bezeichnet den Wert (falls null) und sonst den Zielwert. |

5. Binäre Bäume, ihre Tiefe und die Anzahl ihrer Knoten:

5.1. Wie viele Knoten kann ein bin. Baum der Tiefe 23 höchstens haben (ungefähr)? **8 Millionen**

5.2. Welche Tiefe hat ein bin. Baum mit 10 Tausend Knoten mindestens (genau)? **14**

6. Was ist ein Modul?

Ein Modul ist ein Behälter für Variablen (Attribute), Unterprogramme (Methoden), Typen, Module etc., der aus mindestens zwei Teilen besteht, einem öffentlichen (oder sichtbaren oder ungeschützten) und einem privaten (oder nicht-sichtbaren oder geschützten) Teil. Von außerhalb des Moduls kann man nur auf die Größen im öffentlichen Teil zugreifen, aber nicht auf die Größen im privaten Teil.

Lösung 4 (20 Punkte):

```

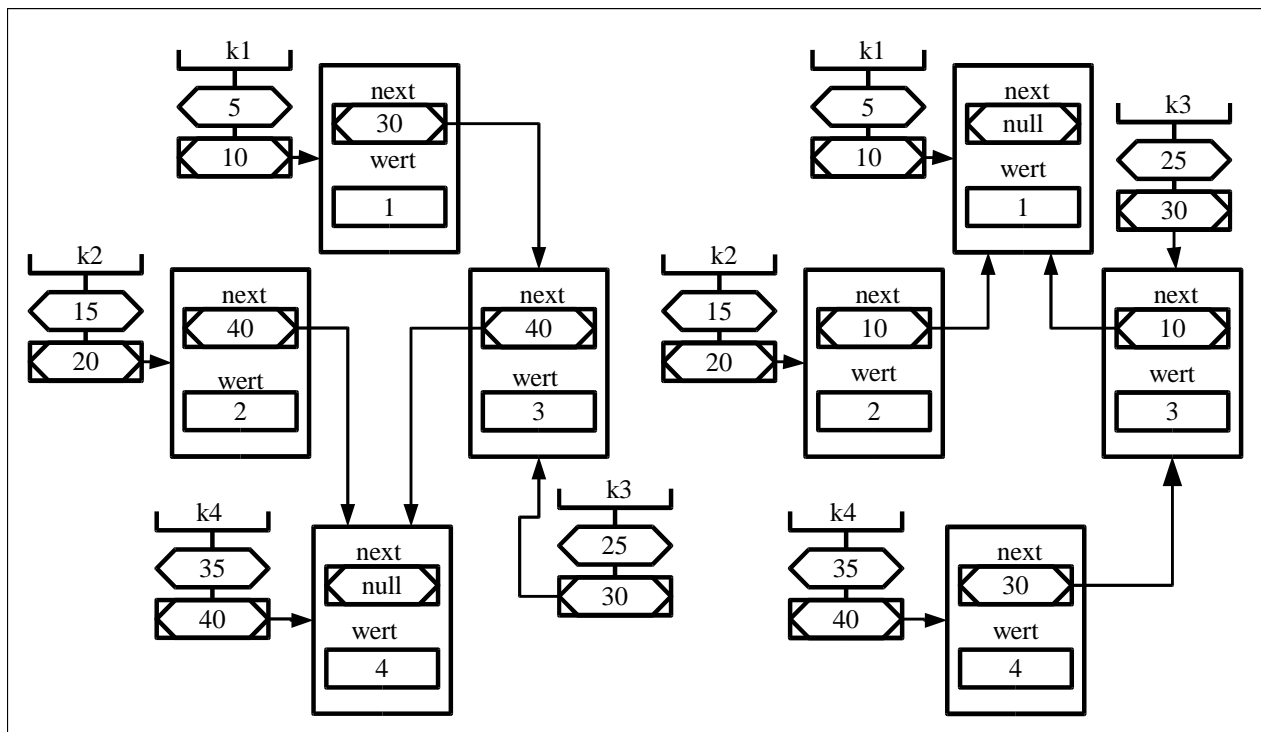
1  static public int anzVokale(ArrayList<String[]> alsr) {
2      // Wie viele grosse Vokale ('A', 'E', 'I', 'O', 'U')
3      // kommen in (den Komponenten der Komponenten von) alsr vor?
4      // Diese Funktion liefert die Antwort.
5
6      int anz = 0;
7      for (String[] sr : alsr) {
8          for (String s : sr) {
9              for (int i=0; i<s.length(); i++) {
10                 char c = s.charAt(i);
11                 if (c=='A' || c=='E' || c=='I' || c=='O' || c=='U') anz++;
12             }
13         }
14     }
15
16     return anz;
17 } // anzVokale

```

Lösung 5 (15 Punkte):

Die Variablen k1 bis k4

wenn der Ausführer die Zeile 15 erreicht, links, und
wenn der Ausführer die Zeile 20 erreicht, rechts:



Lösung 6 (15 Punkte):

```
// 123456789012345 <- Spaltenlineal

printf(" |A %+,8d|\n", 12345); // |A +12.345|
printf(" |B %+,8d|\n", -12345); // |B -12.345|
printf(" |C %-,8d|\n", 12345); // |C 12.345 |
printf(" |D %-,8d|\n", -12345); // |D -12.345 |
printf(" |E %0,8d|\n", 12345); // |E 0012.345|
printf(" |F %0,8d|\n", -12345); // |F -012.345|
printf(" |G %3.5s|\n", 12345); // |G 12345|
printf(" |H %3.5s|\n", -12345); // |H -1234|

// 123456789012345 <- Spaltenlineal
```