

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 4 bis 6 stehen auf der **Rückseite** dieses Blatts!

Aufgabe 1 (20 Punkte): Schreiben Sie eine Methode entsprechend der folgenden Spezifikation:

```
1  static public int[] h2000_2500(int[] jz) {
2      // Wie oft kommt die Jahreszahl 2000 in jz vor? Und die Zahl 2001?
3      // Und die Zahl 2002? ... Und die Zahl 2500? Diese Methode liefert
4      // die Antworten auf diese 501 Fragen in Form einer Reihung der
5      // Laenge 501.
6      // Beispiel:
7      // int ir01 = {2499, 2000, 1999, 2000, 2501, 2000, 2500, 2001, 2500};
8      // Dann ist h2000_2500(ir01) eine Reihung mit den 501 Komponenten
9      // {3, 1, 0, 0, ..., 0, 1, 2}
10     // denn in der Reihung ir01 kommt die Jahreszahl
11     // 2000 genau 3 mal vor, 2001 kommt 1 mal vor, 2499 kommt auch
12     // 1 mal vor und 2500 kommt 2 mal vor. Die Jahreszahlen 1999 und
13     // 2501 werden nicht gezaehlt.
14     ...
15 }
```

Aufgabe 2 (20 Punkte): Schreiben Sie eine **rekursive** Methode entsprechend der folgenden Spezifikation (**iterative** Lösungen sind hier nicht erlaubt!):

```
1  static public int rekuSum(int[] ir, final int IND) {
2      // Liefert die Summe
3      // ir[IND] + ir[IND+1] + ir[IND+2] + ... + ir[ir.length-1]
4      // d.h. die Summe aller Komponenten von ir, deren Index
5      // groesser oder gleich IND ist.
6      // Falls IND kein fuer die Reihung ir geeigneter Index ist,
7      // wird 0 als Ergebnis geliefert.
8      // Beispiel:
9      // int[] ir02 = {10, 20, 30, 40};
10     // rekuSum(ir02, 2) ist gleich 70
11     // rekuSum(ir02, 1) ist gleich 90
12     // rekuSum(ir02, 0) ist gleich 100
13     // rekuSum(ir02, 3) ist gleich 40
14     // rekuSum(ir02, 4) ist gleich 0
15     // rekuSum(ir02, -1) ist gleich 0
16     ...
17 }
```

Aufgabe 3 (15 Punkte): Betrachten Sie die folgenden Befehle (die z.B. in einer main-Method stehen könnten):

```
1  String[] rs = {"ABC", "ABC", new String("ABC"), "", null};
2  Object[] ro = new Object[4];
3  ro[0] = rs;
4  ro[2] = rs;
5  ro[3] = ro;
```

Wie sehen die Reihungsvariablen `rs` und `ro` aus, nachdem die Zeile 5 fertig ausgeführt ist?
Beantworten Sie diese Frage durch eine **Bojen-Darstellung** der beiden Variablen.

Aufgabe 4 (15 Punkte): Diese Aufgabe besteht aus 3 (voneinander unabhängigen) **Teilaufgaben**.

Was geben die folgenden Befehlsfolgen zum Bildschirm aus? Für jede der drei Befehlsfolgen soll Ihre Lösung klar erkennen lassen, wie viele Zeilen und Zeichen (und welche Zeichen) ausgegeben werden.

```

1      // Befehlsfolge 1:
2      int a = +3;
3      int b = -3;
4      for (int i1=-4; i1<=+5; i1=i1+2) {
5          a = (a-b);
6          b = (a+b) / 2;
7          printf("i1: %d, a: %d, b: %d\n", i1, a, b);
8      }
9
10     // Befehlsfolge 2:
11     int[] ir = {30, 20, 10};
12     pln(Arrays.toString(ir));
13     for (int n : ir) n++;
14     pln(Arrays.toString(ir));
15
16     // Befehlsfolge 3
17     int[] jr = {17, 22, 35, 49, 54, 61, 76};
18     int s = 37;
19
20     int i = jr.length / 2;
21     int d = i + 1;
22     while (true) {
23         printf("jr[%d]: %d\n", i, jr[i]);
24         d = d / 2;
25         if (s == jr[i] || d == 0) break;
26         i = i + (s < jr[i] ? -d : +d);
27     }

```

Aufgabe 5 (15 Punkte): Das folgende Java-Programm besteht aus den 2 Klassen KA und KB. In der main-Methode sind fünf Zeilen besonders gekennzeichnet (als Zeile 01, Zeile 02, ...):

```

public class KA {
    // -----
    static int a01 = 101;
    int a02 = 102;
    // -----
    static public void main(String[] _) {
        pln("-----"); // Zeile 01
        Class<KB> oa = KB.class; // Zeile 02
        KB ob = new KB(); // Zeile 03
        KB oc = new KB(); // Zeile 04
        KA od = new KA(); // Zeile 05
        pln("-----");
    }
    // -----
    static void pln(Object ob) {System.out.println(ob);}
    // -----
} // class KA

class KB {
    static int b01 = 201;
    static int b02 = 202;
    int b03 = 203;
    int b04 = 204;
} // class KB

```

- 5.1. Welche int-Variablen existieren nach Ausführung von Zeile 01?
- 5.2. Welche int-Variablen sind nach Ausführung von Zeile 02 hinzugekommen?
- 5.3. Welche int-Variablen sind nach Ausführung von Zeile 03 hinzugekommen?
- 5.4. Welche int-Variablen sind nach Ausführung von Zeile 04 hinzugekommen?
- 5.5. Welche int-Variablen sind nach Ausführung von Zeile 05 hinzugekommen?

Geben Sie von jeder Variablen den *vollen Namen* an (d.h. einen Namen der Form Modul . Name).

Aufgabe 6: (15 Punkte) Beantworten Sie die folgenden Fragen möglichst *kurz*, aber *genau* und benutzen Sie dabei die im seminaristischen Unterricht behandelten *Fachbegriffe*:

Zur Erinnerung: Integer ist ein Untertyp von Number.

1. Was ist der Typ `Collection<Number>` relativ zum Typ `Collection<Integer>`?

Ein Obertyp oder ein Untertyp oder weder-noch?

2. Was ist der Typ `Number[]` relativ zum Typ `Integer[]`?

Ein Obertyp oder ein Untertyp oder weder-noch?

3. Was ist (in Java) eine *Sammlung* (engl. collection) und was ist ein *Behälter* (engl. container)? Geben Sie möglichst die *inhaltlichen Definitionen* an, die im SU behandelt wurden (nicht die *formalen Definitionen*, die auch behandelt wurden).

4. Beschreiben Sie (ganz kurz) ein Problem, welches man *nicht* mit einer for-each-Schleife lösen kann, wohl aber mit einer for-i-Schleife.

5. Beschreiben Sie (ganz kurz) einen Anwendungsfall, in dem man *reproduzierbare* Zufallswerte (im Gegensatz zu *nicht-reproduzierbaren* Zufallswerten) verwendet.

6. Welche 3 grundlegenden Eigenschaften von Objekten der Stromklasse `CharArrayReader` kann man aus dem Klassen-Namen `CharArrayReader` schließen?

7. Warum ist die Klasse `Class` *generisch* mit *einem* Typparameter `T` (`Class<T>`)?

8. Wie kann die Vereinbarung einer *paketweit erreichbaren Klassenschnittstelle* aussehen? Geben Sie ein Beispiel für eine solche Vereinbarung an. Alle hier nicht festgelegten Einzelheiten können Sie so (einfach) gestalten, wie Sie möchten, aber Ihr Beispiel muss vom Java-Ausführer akzeptiert werden.

**Beurteilung dieser Klausur und
Gesamt-Note:**

A1	
A2	
A3	
A4	
A5	
A6	
Summe	
Klausur-Note	
Übungs-Note	
Gesamt-Note für MB2-PR2:	
Datum	11.07.12

**Korrigierte Beurteilung dieser Klausur und
Gesamt-Note:**

A1	
A2	
A3	
A4	
A5	
A6	
Summe	
Klausur-Note	
Übungs-Note	
Gesamt-Note für MB2-PR2	
Datum	

Lösung 1 (20 Punkte): Zwei voneinander unabhängige Teilaufgaben. Beide müssen rekursiv gelöst werden (sie dürfen keine Schleifen enthalten).

```

1  static public int[] h2000_2015(int[] jz) {
2      // Wie oft kommt die Jahreszahl 2000 in jz vor? Und die Zahl 2001?
3      // Und die Zahl 2002? ... Und die Zahl 2015? Diese Methode liefert
4      // die Antworten auf diese 16 Fragen in Form einer der Reihung der
5      // Laenge 16.
6      // Beispiel:
7      // int ir01 = {2014, 2000, 1999, 2000, 2016, 2000, 2015, 2001, 2015};
8      // Dann ist h200_2015(ir01) eine Reihung mit den 16 Komponenten
9      // {3, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 2}
10     // denn in der Reihung ir01 kommt die Jahreszahl
11     // 2000 genau 3 mal vor, 2001 kommt 1 mal vor, 2014 kommt auch
12     // 1 mal vor und 2015 kommt 2 mal vor. Die Jahreszahlen 1999 und
13     // 2016 werden nicht gezaehlt.
14
15     int[] erg = new int[501];
16
17     for (int z : jz) {
18         if (2000 <= z && z <= 2500) {
19             erg[z-2000]++;
20         }
21     }
22
23     return erg;
24 }

```

Lösung 2 (20 Punkte):

```

1  static public int rekuSum(int[] ir, final int IND) {
2      // Liefert die Summe
3      // ir[IND] + ir[IND+1] + ir[IND+2] + ... + ir[ir.length-1]
4      // d.h. die Summe aller Komponenten von ir, deren Index
5      // groesser oder gleich IND ist.
6      // Falls IND kein fuer die Reihung ir geeigneter Index ist,
7      // wird 0 als Ergebnis geliefert.
8      // Beispiel:
9      // int[] ir02 = {10, 20, 30, 40};
10     // rekuSum(ir02, 2) ist gleich 70
11     // rekuSum(ir02, 1) ist gleich 90
12     // rekuSum(ir02, 0) ist gleich 100
13     // rekuSum(ir02, 3) ist gleich 40
14     // rekuSum(ir02, 4) ist gleich 0
15     // rekuSum(ir02, -1) ist gleich 0
16
17     // Einfache Faelle:
18     if (IND < 0 || ir.length <= IND) return 0;
19
20     // Rekursiver Fall:
21     return ir[IND] + rekuSum(ir, IND+1);
22 }

```

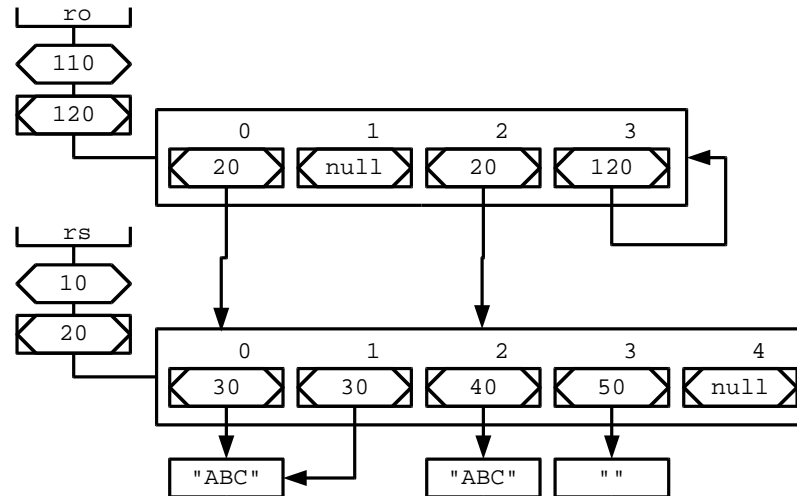
Lösung 3 (15 Punkte) : Betrachten Sie die folgenden Befehle (die z.B. in einer main-Method stehen könnten):

```

1  String[] rs = {"ABC", "ABC", new String("ABC"), "", null};
2  Object[] ro = new Object[4];
3  ro[0] = rs;
4  ro[2] = rs;
5  ro[3] = ro;

```

Wie sehen die Reihungsvariablen `rs` und `ro` aus, nachdem die Zeile 5 fertig ausgeführt ist? Beantworten Sie diese Frage durch eine *Bojen-Darstellung* der beiden Variablen.



Lösung 4 (15 Punkte): Diese Aufgabe besteht aus 3 (voneinander unabhängigen) Teilaufgaben. Was geben die folgenden Befehlsfolgen zum Bildschirm aus? Für jede der drei Befehlsfolgen soll Ihre Lösung klar erkennen lassen, wie viele Zeilen und Zeichen (und welche Zeichen) ausgegeben werden.

Befehlsfolge 1:

```
i1: -4, a: +6, b: +1
i1: -2, a: +5, b: +3
i1: +0, a: +2, b: +2
i1: +2, a: +0, b: +1
i1: +4, a: -1, b: +0
```

Befehlsfolge 2

```
[30, 20, 10]
[30, 20, 10]
```

Befehlsfolge 3

```
jr[3]: 49
jr[1]: 22
jr[2]: 35
```

Lösung 5 (15 Punkte): Das folgende Java-Programm besteht aus den 2 Klassen KA und KB. In der main-Methode sind fünf Zeilen besonders gekennzeichnet (als Zeile 01, Zeile 02, ...):

```
public class KA {
    // -----
    static int a01 = 101;
        int a02 = 102;
    // -----
    static public void main(String[] _) {
        pln("-----"); // Zeile 01
        Class<KB> oa = KB.class; // Zeile 02
        KB ob = new KB(); // Zeile 03
        KB oc = new KB(); // Zeile 04
        KA od = new KA(); // Zeile 05
        pln("-----");
    }
    // -----
    static void pln(Object ob) {System.out.println(ob);}
    // -----
} // class KA

class KB {
    static int b01 = 201;
    static int b02 = 202;
        int b03 = 203;
        int b04 = 204;
} // class KB
```

- 5.1. Welche int-Variablen existieren nach Ausführung von Zeile 01 ?
- 5.2. Welche int-Variablen sind nach Ausführung von Zeile 02 hinzugekommen?
- 5.3. Welche int-Variablen sind nach Ausführung von Zeile 03 hinzugekommen?
- 5.4. Welche int-Variablen sind nach Ausführung von Zeile 04 hinzugekommen?
- 5.5. Welche int-Variablen sind nach Ausführung von Zeile 05 hinzugekommen?

Geben Sie von jeder Variablen den *vollen Namen* an (d.h. einen Namen der Form Modul.Name).

- 5.1. Nach Zeile 01: **KA.a01**
- 5.2. Nach Zeile 02: **KB.b01, KB.b02**
- 5.3. Nach Zeile 03: **ob.b03, ob.b04**
- 5.4. Nach Zeile 04: **oc.b03, oc.b04**
- 5.5. Nach Zeile 05: **od.a02**

Lösung 6 (15 Punkte): Beantworten Sie die folgenden Fragen möglichst *kurz*, aber *genau* und benutzen Sie dabei die im seminaristischen Unterricht behandelten *Fachbegriffe*:

Zur Erinnerung: Integer ist ein Untertyp von Number.

1. Was ist der Typ `Collection<Number>` relativ zum Typ `Collection<Integer>`?

Ein Obertyp oder ein Untertyp oder weder-noch?

Weder-noch

2. Was ist der Typ `Number[]` relativ zum Typ `Integer[]`?

Ein Obertyp oder ein Untertyp oder weder-noch?

Ein Obertyp

3. Was ist (in Java) eine *Sammlung* (engl. collection) und was ist ein *Behälter* (engl. container)? Geben Sie möglichst die *inhaltlichen Definitionen* an, die im SU behandelt wurden (nicht die *formalen Definitionen*, die auch behandelt wurden).

Sammlung:

Ein Objekt, in dem man Objekte sammeln kann (d.h. einfügen, darin suchen, entfernen)

Behälter:

Ein Grabo-Objekt, in das man Grabo-Objekte hineintun kann.

4. Beschreiben Sie (ganz kurz) ein Problem, welches man *nicht* mit einer for-each-Schleife lösen kann, wohl aber mit einer for-i-Schleife.

Zwei Reihungen Komponente-für-Komponente miteinander vergleichen.

Die Werte der Komponenten einer Reihung oder Sammlung verändern.

...

5. Beschreiben Sie (ganz kurz) einen Anwendungsfall, in dem man *reproduzierbare* Zufallswerte (im Gegensatz zu *nicht-reproduzierbaren* Zufallswerten) verwendet.

Erzeugung von Testdaten

6. Welche 3 grundlegenden Eigenschaften von Objekten der Stromklasse `CharArrayReader` kann man aus dem Klassen-Namen `CharArrayReader` schließen?

- Eingabestrom

- Zeichenorientiert

- Kann "angeschraubt werden" an eine Reihung von `char` (d.h. an ein Objekt des Typs `char[]`).

7. Warum ist die Klasse `Class` *generisch* mit *einem* Typparameter `T` (`Class<T>`)?

Damit sie zwei Methoden (namens `cast` und `newInstance`) mit dem Rückgabotyp `T` haben kann.

8. Wie kann die Vereinbarung einer *paketweit erreichbaren Klassenschnittstelle* aussehen? Geben Sie ein Beispiel für eine solche Vereinbarung an. Alle hier nicht festgelegten Einzelheiten können Sie so (einfach) gestalten, wie Sie möchten, aber Ihr Beispiel muss vom Java-Ausführer akzeptiert werden.

```
static interface Anschraubbar {}
```