

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 Blattes!

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

```
1  static int anz7erZiffern(long n) {
2      // Wie viele Ziffern braucht man, um n als 7-er-Zahl (als Zahl im
3      // System mit der Basis 7) darzustellen? Diese Funktion liefert
4      // die Antwort.
5      //
6      // Beispiele:
7      // n als 10-er-Zahl    n als 7-er-Zahl    anz7erZiffern(n)
8      //      0              0                  1
9      //      3              3                  1
10     //     -3             -3                  1
11     //      7             10                  2
12     //     -10            -13                  2
13     //      48             66                  2
14     //      49             100                 3
15     // ...
16 } // anz7erZiffern
```

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

```
1  static int wieOft(ArrayList<String[]> alsr) {
2      // Wie oft kommt der String "AB" in
3      // (den Komponenten der Komponenten von) alsr vor?
4      // Diese Funktion liefert die Antwort. Zwei Beispiele:
5      // String[] sr1 = {"ABCAB", "XABX", "ABABAB"}; // 6 mal "AB"
6      // String[] sr2 = {"X", "AB", ""};           // 1 mal "AB"
7      // String[] sr3 = {};                         // 0 mal "AB"
8      //
9      // ArrayList<String[]> alsrA = new ArrayList<String[]>();
10     // ArrayList<String[]> alsrB = new ArrayList<String[]>();
11     //
12     // alsrA.add(sr1);
13     // alsrA.add(sr2);
14     // alsrA.add(sr3);
15     //
16     // wieOft(alsrA) ist gleich 7
17     // wieOft(alsrB) ist gleich 0
18     // ...
19 } // wieOft
```

**Aufgabe 3 (15 Punkte):** Vereinbaren Sie eine Klasse namens `MeinFenster` als Erweiterung der Klasse `JFrame`.

Wenn ein Fenster des Typs `MeinFenster` minimiert wird (d.h. wenn ein `MeinFenster`-Objekt ein Ereignis der Art `windowIconified` erzeugt), soll die Meldung "Bin jetzt eine Ikone!" zur Standardausgabe ausgegeben werden. Wenn das dritte solche Ereignis erzeugt wird (d.h. wenn das Fenster zum dritten Mal minimiert wird) soll das umgebende Programm beendet werden (mit der Anweisung `System.exit(17);`).

**Zur Erinnerung:** Die Ereignisart `windowIconified` gehört zur Ereignisoberart *Fensterereignis*. Dieser Oberart ist die Schnittstelle `WindowListener` zugeordnet.

Beim Erzeugen eines `MeinFenster`-Objekts soll es möglich sein, einen *Titel* anzugeben, d.h. einen String wie z.B. "Mein erster Rahmen", der dann im Rahmen des Fensters erscheint.

`MeinFenster`-Objekte sollen als Fenster der folgenden *Größe* dargestellt werden: 200 Pixel in x-Richtung und 100 Pixel in y-Richtung.

Vergessen Sie nicht, die Fenster sichtbar zu machen.

**Aufgabe 4 (15 Punkte)** Betrachten Sie die folgenden drei Codestücke:

```

1      // Codestueck A:
2      int a1 = 2;
3      int a2 = -3;
4      int a3 = 4;
5      for (int a4=0; a4<=16; a4+=3) {
6          a1 = a1 + a2;
7          a2 = a2 * -1;
8          a3 = a1 - a3 + a4;
9          pln("a3: " + a3);
10     }
11
12     // Codestueck B:
13     String s = "5X4X3X2X1X";
14     int n = -1;
15     while (true) {
16         n = s.indexOf('X', n+1);
17         if (n== -1) break;
18         printf("Ab %d: %s%n", n, s.substring(n));
19     }
20
21     // Codestueck C:
22     final int MAX = 4;
23     for (int i=1; i<=MAX; i++) {
24         p(i + " ");
25         for (int j=0; j<MAX-i; j++) p(' ');
26         for (int j=0; j<i; j++) p('X');
27         for (int j=0; j<i; j++) p('Y');
28         pln();
29     }

```

Geben Sie für jedes Codestück an, was es (zur Standardausgabe) ausgibt. Schreiben Sie dabei *besonders deutlich*, so dass klar zu erkennen ist, wie viele *Zeilen*, wie viele *Zeichen* und *welche* Zeichen ausgegeben werden (in Zweifelsfällen sollten Sie ihre Lösung noch mal sauber abschreiben).

**Aufgabe 5 (15 Punkte):** Betrachten Sie die folgende Befehlsfolge:

```
1  int[]    ri = {10, 20};
2  double[] rd = {};
3  String   st = "Hallo";
4  Object[] ro = {rd, ri, st, "Hallo", null};
```

Stellen Sie die vier Variablen `ri`, `rd`, `st` und `ro` als Bojen dar.

**Aufgabe 6: (15 Punkte)** Beantworten Sie die folgenden Fragen möglichst kurz, aber genau. Benutzen Sie dabei möglichst die in der Vorlesung eingeführten Fachbegriffe.

1. Was ist ein *Modul*?

2. Geben Sie von jedem der folgenden Befehle an, zu welcher *Art* von Befehlen er gehört:

```
1  class Anna extends Berta {int n = 17;}
2  float felix = 2.5F;
3  ... "Ergebnis: " + (2.0F * felix - 1.0F) ...
4  println("Ergebnis: " + (2.0F * felix - 1.0F));
```

3. Aus *wie vielen* Teilen besteht eine *Variable* höchstens, und wie *heißen* diese Teile?

4. Beschreiben Sie (möglichst kurz) zwei unterschiedliche Probleme, die man mit einer for-i-Schleife, aber *nicht mit einer einer for-each-Schleife* lösen kann.

5. Was macht der *erste Befehl* in jedem Konstruktor?

6. Woraus besteht der *volle Name* einer Klasse `k1` die zu einem Paket `p1` gehört?

**Achtung:** `k1` und `p1` sollen hier keine Namen (einer Klasse bzw. eines Pakets) sein, sondern *Bezeichnungen*, die unabhängig von Namen sind, so dass z.B. folgende Aussagen sinnvoll sind:

"Die Klasse `k1` könnte den Namen "Otto!" haben" oder

"Der Name von `p1` kann z.B. mit dem Buchstaben 'j' beginnen" etc.

7. Welche Frage stellt sich der Java-Ausführer immer, wenn er gerade eine Ausnahme `a` geworfen hat?

8. In Java gibt es etwa 70 *Ereignisoberarten* (wie z.B. Fensterereignis, Mausereignis, Mausbewegungsereignis, Mausradereignis, Aktionsereignis etc.). Zu welchen Ereignisoberarten gibt es *keine* Adapter-Klasse?

**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): Schreiben Sie eine Methode entsprechend der folgenden Spezifikation:**

```

1  static int anz7erZiffern(long n) {
2      // Wie viele Ziffern braucht man, um n als 7-er-Zahl (als Zahl im
3      // System mit der Basis 7) darzustellen? Diese Funktion liefert
4      // die Antwort.
5      //
6      // Beispiele:
7      // n als 10-er-Zahl    n als 7-er-Zahl    anz7erZiffern(n)
8      //      0              0                  1
9      //      3              3                  1
10     //     -3             -3                  1
11     //      7             10                  2
12     //     -10            -13                  2
13     //      48             66                  2
14     //      49             100                 3
15
16     n = Math.abs(n); // oder: if (n<0) n = -n;
17     int erg = 1;
18
19     while (true) {
20         if (n<7) break;
21         n = n/7;
22         erg++;
23     }
24     return erg;
25
26 } // anz7erZiffern

```

**Lösung 2 (20 Punkte): Schreiben Sie eine Methode, die der folgenden Spezifikation entspricht:**

```

1  static int wieOft(ArrayList<String[]> alsr) {
2      // Wie oft kommt der String "AB" in
3      // (den Komponenten der Komponenten von) alsr vor?
4      // Diese Funktion liefert die Antwort. Zwei Beispiele:
5      //
6      // String[] sr1 = {"ABCAB", "XABX", "ABABAB"}; // 6 mal "AB"
7      // String[] sr2 = {"X", "AB", ""}; // 1 mal "AB"
8      // String[] sr3 = {}; // 0 mal "AB"
9      //
10     // ArrayList<String[]> alsrA = new ArrayList<String[]>();
11     // ArrayList<String[]> alsrB = new ArrayList<String[]>();
12     //
13     // alsrA.add(sr1);
14     // alsrA.add(sr2);
15     // alsrA.add(sr3);
16     //
17     // wieOft(alsrA) ist gleich 7
18     // wieOft(alsrB) ist gleich 0
19
20     int erg = 0;
21
22     for (String[] sr : alsr) {
23         for (String s : sr) {
24             for (int i=0; i<s.length()-1; i++) {
25                 if ('A'==s.charAt(i) && 'B'==s.charAt(i+1)) {
26                     erg++;
27                     i++;
28                 }
29             }
30         }
31     }
32
33     return erg;
34 } // wieOft

```

**Lösung 3 (15 Punkte):** Vereinbaren Sie eine Klasse namens `MeinFenster` als Erweiterung der Klasse `JFrame`.

Wenn ein Fenster des Typs `MeinFenster` minimiert wird (d.h. wenn ein `MeinFenster`-Objekt ein Ereignis der Art `windowIconified` erzeugt), soll die Meldung "Bin jetzt eine Ikone!" zur Standardausgabe ausgegeben werden. Wenn das dritte solche Ereignis erzeugt wird (d.h. wenn das Fenster zum dritten Mal minimiert wird) soll das umgebende Programm beendet werden (mit der Anweisung `System.exit(1);`).

**Zur Erinnerung:** Die Ereignisart `windowIconified` gehört zur Ereignisoberart *Fensterereignis*. Dieser Oberart ist die Schnittstelle `WindowListener` zugeordnet.

Beim Erzeugen eines `MeinFenster`-Objekts soll es möglich sein, einen *Titel* anzugeben, d.h. einen String wie z.B. "Mein erster Rahmen", der dann im Rahmen des Fensters erscheint.

`MeinFenster`-Objekte sollen als Fenster der folgenden *Größe* dargestellt werden: 200 Pixel in x-Richtung und 100 Pixel in y-Richtung.

Vergessen Sie nicht, die Fenster sichtbar zu machen.

```

1 class MeinFenster extends JFrame {
2     static class MeinFensterBehandler extends WindowAdapter {
3         int anz = 0;
4         @Override
5         public void windowIconified(WindowEvent we) {
6             System.out.println("Bin jetzt eine Ikone!");
7             if (++anz == 3) System.exit(1);
8         }
9     } // class MeinFensterBehandler
10
11     MeinFensterBehandler meinFensterBehandler = new MeinFensterBehandler();
12
13     public MeinFenster(String titel) {
14         super(titel);
15         addWindowListener(meinFensterBehandler);
16         setBounds(600, 800, 200, 100);
17         setVisible(true);
18     }
19
20 } // class MeinFenster

```

**Lösung 4 (15 Punkte):** Geben Sie für jedes Codestück an, was es (zur Standardausgabe) ausgibt.

-----  
**Codestueck A:**

```

a3: -5
a3: 10
a3: -5
a3: 16
a3: -5
a3: 22

```

-----  
**Codestueck B:**

```

Ab 1: X4X3X2X1X
Ab 3: X3X2X1X
Ab 5: X2X1X
Ab 7: X1X
Ab 9: X

```

-----  
**Codestueck C:**

```

1   XY
2  XXY
3 XXXYY
4 XXXYYY

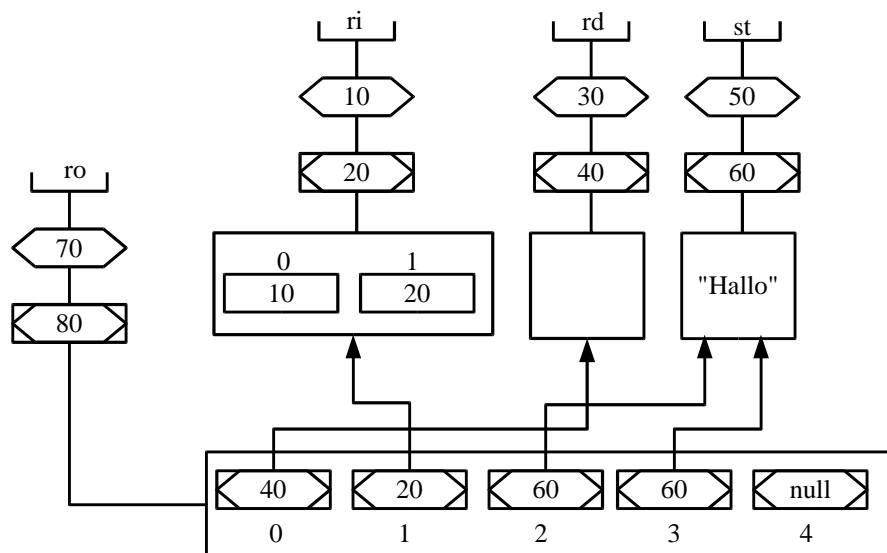
```

-----

**Lösung 5 (15 Punkte):** Betrachten Sie die folgende Befehlsfolge:

```
1  int[]    ri = {10, 20};  
2  double[] rd = {};  
3  String   st = "Hallo";  
4  Object[] ro = {rd, ri, st, "Hallo", null};
```

Stellen Sie die vier Variablen **ri**, **rd**, **st** und **ro** als Bojen dar.



**Lösung 6:** (15 Punkte) Beantworten Sie die folgenden Fragen möglichst kurz, aber genau. Benutzen Sie dabei möglichst die in der Vorlesung eingeführten Fachbegriffe.

1. Was ist ein *Modul*?

Ein Behälter für Variablen (oder: Attribute), Unterprogramme (oder: Methoden), Typen, Module etc., der aus mindestens 2 Teilen besteht: Einem öffentlichen Teil und einem privaten Teil.

2. Geben Sie von jedem der folgenden Befehle an, zu welcher *Art* von Befehlen er gehört:

```
1  class Anna extends Berta {int n = 17;}           // (Klassen-) Vereinbarung
2  float felix = 2.5F;                               // (Variablen-) Vereinbarung
3  ... "Ergebnis: " + (2.0F * felix - 1.0F) ...      // Ausdruck
4  println("Ergebnis: " + (2.0F * felix - 1.0F));    // Anweisung
```

3. Aus *wie vielen* Teilen besteht eine *Variable* höchstens, und wie *heißen* diese Teile?

Aus 4 Teilen: Name, Referenz, Wert, Zielwert

4. Beschreiben Sie (möglichst kurz) zwei unterschiedliche Probleme, die man mit einer for-i-Schleife, aber *nicht mit einer einer for-each-Schleife* lösen kann.

- Wir haben eine Reihung und wollen die Werte der Komponenten verändern (z.B. alle auf 0 setzen).

- Wir wollen zwei Reihungen Komponente für Komponente vergleichen

5. Was macht der *erste Befehl* in jedem Konstruktor?

Er ruft einen Konstruktor der direkten Oberklasse auf.

6. Woraus besteht der *volle Name* einer Klasse k1 die zu einem Paket p1 gehört?

Achtung: k1 und p1 sollen hier keine Namen (einer Klasse bzw. eines Pakets) sein, sondern *Bezeichnungen*, die unabhängig von Namen sind, so dass z.B. folgende Aussagen sinnvoll sind:

"Die Klasse k1 könnte den Namen "Otto!" haben" oder

"Der Name von p1 kann z.B. mit dem Buchstaben 'j' beginnen" etc.

Der volle Name von k1 besteht aus dem vollen Namen von p1 gefolgt von einem Punkt '.' und dem (einfachen) Namen von k1.

7. Welche Frage stellt sich der Java-Ausführer immer, wenn er gerade eine Ausnahme a geworfen hat?

Trat die Ausnahme a in einem try-Block auf, dem ein zum Fangen von a geeigneter catch-Block folgt?

8. In Java gibt es etwa 70 *Ereignisoberarten* (wie z.B. Fensterereignis, Mausereignis, Mausbewegungsereignis, Mausradereignis, Aktionsereignis etc.). Zu welchen Ereignisoberarten gibt es *keine* Adapter-Klasse?

Zu den Ereignisoberarten, zu denen nur eine (einzige) Ereignisart gehört (z.B. Mausradereignis, Aktionsereignis etc.)