

Vorname

Nachname

Matrikel-Nr

Tragen Sie Ihre Lösungen zu den **Aufgaben 1 bis 3** in das folgende **Formular** ein. Schreiben Sie Ihre Lösungen zu **Aufgabe 4** und **Aufgabe 5** auf extra Blätter und kennzeichnen Sie diese Blätter oben links mit Ihrem **Nachnamen**.

Formular für die **Lösungen** zu den Aufgaben 1 bis 3:

1.1. Anzahl der Elemente?	2. Die Nummern der falschen Zeilen von Claudia:
1.2. Anzahl der Klassenelemente?	3.1. Die Ausgabe der Prozedur a31:
1.3. Anzahl der Instanzelemente?	3.2. Die Ausgabe der Prozedur a32:
1.4. Anzahl und Namen der Module?	3.3. Die Ausgabe der Prozedur a33:
	3.4. Die Ausgabe der Prozedur a34:
1.5. Die vollen Namen aller Elemente?	
1.6. Ausgabe des Programms Betrag?	

Aufgabe 1 (18 Punkte) : Betrachten Sie das folgende Programm und beantworten Sie dann die darunterstehenden Fragen:

```

1 public class Betrag {
2     static private int anzahl0b;
3     static public int getAnzahl0b() { return anzahl0b;}
4
5     private long betrag;
6     public long getBetrag() { return betrag;}
7     public void setBetrag(long neu) {
8         if (neu >= 0) {
9             betrag = neu;
10        } else {
11            betrag = 0;
12            System.out.println(neu + " ist negativ!");
13        }
14    } // setBetrag

```

```
15
16  Betrag(long neu) {
17      setBetrag(neu);
18      anzahlOb++;
19  } // Konstruktor Betrag
20
21  static public void main(String[] args) {
22      Betrag betrag1 = new Betrag(-15);
23      Betrag betrag2 = new Betrag(250);
24      betrag1.setBetrag(betrag2.getBetrag());
25      System.out.println("Anzahl Objekte: " + anzahlOb);
26      System.out.println("Betrag1:      " + betrag1.getBetrag());
27  } // main
28 } // class Betrag
```

Frage 1.1.: Wieviele **Elemente** enthält die Klasse **Betrag**?

Frage 1.2.: Wieviele **Klassenelemente** enthält die Klasse **Betrag**?

Frage 1.3.: Wieviele **Instanzelemente** enthält die Klasse **Betrag**?

Frage 1.4.: Wenn der Ausführer das Programm **Betrag** bis Zeile 25 ausgeführt hat, **wieviele** Module existieren dann und **wie heißen** diese Module?

Frage 1.5.: Wie heißen die **Elemente** in all diesen Modulen mit **vollem Namen**? Der volle Name eines Elements besteht aus dem Namen des betreffenden **Moduls**, einem **Punkt** und dem Namen des betreffenden **Elementes**, z.B. **otto.emil** für ein Element namens **emil** in einem Modul namens **otto**.

Frage 1.6.: Was gibt das Programm **Betrag** zum Bildschirm aus?

Aufgabe 2 (18 Punkte): Die folgende Klasse enthält mehrere **falsche Zeilen**, die den Compiler zu einer Fehlermeldung der Art "**Can't make a static reference to ...**" veranlassen. Welche Zeilen sind das? Geben Sie als Lösung nur die **Zeilennummern** der falschen Zeilen an.

```
1 public class Claudia {
2
3     boolean b1 = true;
4     boolean b2 = b1;
5     static boolean b3 = b2;
6     static boolean b4 = false;
7
8     static double d1;
9     double d2 = d1;
10
11     static boolean f1(boolean b) {return b1 && b; }
12
13     static boolean f2(boolean b) {return b4 || b; }
14
15     static boolean f3(boolean b) {return f2(b); }
16
17     static boolean f4(boolean b) {return f5(2.7); }
18
19     boolean f5(double d) {return d1 < d; }
20
21     boolean f6(double d) {return d2 >= d; }
22
23     boolean f7(double d) {return f3(false);}
24
25     boolean f8(double d) {return f5(d); }
26
27 } // class Claudia
```

Aufgabe 3 (18 Punkte): Jede der folgenden Prozeduren gibt eine Zeile zum Bildschirm aus. Geben Sie für jede Prozedur an, wie diese Zeile aussieht.

```
1  // -----
2  static void a31() {
3      int a1 = 17;
4      int a2 = 2 * a1;
5      a1 += 3;
6      int a3 = 2 * a1++;
7      a1 = a3 + 5;
8      a3 = a1 + a3;
9      System.out.println("a3: " + a3);
10 } // a31
11 // -----
12 static void a32() {
13     int b1 = 10;
14     for (int i=-3; i<0; i++) {
15         b1 += i;
16     }
17     System.out.println("b1: " + b1);
18 } // a32
19 // -----
20 static void a33() {
21     int[] ir = {5, 3, 7, 1};
22     for (int i=0; i<ir.length-1; i++) {
23         ir[i+1] += ir[i] + ir.length;
24     }
25     System.out.print("ir: ");
26     for (int i=ir.length-1; i>=0; i--) {
27         System.out.print(ir[i] + ", ");
28     }
29     System.out.println();
30 } // a33
31 // -----
32 static void a34() {
33     int sum = 0;
34     for (int i=0; i<=3; i++) {
35         switch (i) {
36             case 1: sum = sum + 1;
37             case 3: sum = sum + 3;
38             case 2: sum = sum + 2; break;
39             default: sum = 0; break;
40         }
41     }
42     System.out.println("sum: " + sum);
43 } // a34
44 // -----
```

Aufgabe 4 (18 Punkte): Schreiben Sie eine Funktion namens **anzahl**, die dem folgenden "Skelett" entspricht:

```
1  int anzahl(char c, String[] r) {
2      // Wie oft kommt das Zeichen c in den String-Komponenten der Reihung
3      // r vor? Diese Funktion liefert die Antwort. Falls r eine null-Refe-
4      // renz ist oder auf eine leere Reihung zeigt, wird 0 als Ergebnis
5      // geliefert.
6      ...
7  } // anzahl
```

Aufgabe 5 (18 Punkte): Ein **abgeschlossenes Intervall** auf der Zahlengerade kann man auf verschiedene Weisen beschreiben, unter anderem durch seinen **Anfangspunkt** und seine **Länge**. Zum abgeschlossenen Intervall mit dem Anfangspunkt 2.0 und der Länge 7.0 gehören alle Zahlen zwischen 2.0 und 9.0 (einschliesslich der Endpunkte) und zum Intervall mit dem Anfangspunkt 5.0 und der Länge 3.0 gehören alle Zahlen zwischen 5.0 und 8.0 etc.. **Zwei** solche Intervalle sind entweder **disjunkt** (wenn sie **keinen** gemeinsamen Punkt haben) oder sie "**überlappen sich**" (wenn sie mindestens **einen** gemeinsamen Punkt haben). Schreiben Sie eine Funktion namens **sindDisjunkt**, die dem folgenden "Skelett" entspricht und erfüllen Sie dabei die folgende

Zusatzforderung: In Ihrer Lösung darf **keine einzige if-Anweisung** und auch **keine** andere zusammengesetzte Anweisung (**switch**-, **while**-, **do-while**- oder **for**-Anweisung) vorkommen.

```
1  static boolean sindDisjunkt(double anfang1, double laenge1,
2                               double anfang2, double laenge2) {
3      // Die 4 Parameter beschreiben zwei abgeschlossene Intervalle auf der
4      // Zahlengerade. Diese Funktion liefert true, wenn diese beiden Ab-
5      // schnitte *disjunkt* sind (d.h. wenn sie keinen Punkt gemeinsam
6      // haben). Sonst liefert sie false. Beispiele:
7      // sindDisjunkt(2.0, 7.0, 3.0, 4.0) liefert false
8      // sindDisjunkt(5.0, 3.0, 7.0, 4.0) liefert false
9      // sindDisjunkt(5.0, 3.0, 8.0, 2.0) liefert false (gemeins. Punkt: 8.0)
10     // sindDisjunkt(5.0, 3.0, 8.1, 6.0) liefert true
11     // sindDisjunkt(8.1, 6.0, 5.0, 3.0) liefert true
12     ...
13 }
```

Hinweis 1: Am Anfang des Semesters haben Sie eine Aufgabe gelöst, in der es um **Dreiseite** und ihre Eigenschaften (**rechtwinklig**, **gleichseitig**, **gleichschenkelig** etc.) ging. Lösen Sie die vorliegende Aufgabe 5 "im gleichen Stil wie die Dreiseit-Hausaufgabe".

Hinweis 2: Beim Schreiben der Funktion **sindDisjunkt()** dürfen Sie sich darauf verlassen, dass die Parameter **laenge1** und **laenge2** Werte haben, die grösser als 0 sind.

Hinweis 3: Wundern Sie sich nicht, wenn Ihre Lösung der Aufgabe **sehr kurz** ist.

Formular mit den **Lösungen** zu den Aufgaben 1 bis 3:

1.1. Anzahl der Elemente? 6 Elemente	2. Die Nummern der falschen Zeilen von Claudia: 5, 11, 17
1.2. Anzahl der Klassenelemente? 3 Klassenelemente	3.1. Die Ausgabe der Prozedur a31: a3: 85
1.3. Anzahl der Instanzelemente? 3 Instanzelemente	3.2. Die Ausgabe der Prozedur a32: b1: 4
1.4. Anzahl und Namen der Module? 3 Module: Betrag, betrag1 und betrag2	3.3. Die Ausgabe der Prozedur a33: ir: 28, 23, 12, 5,
	3.4. Die Ausgabe der Prozedur a34: sum: 13
1.5. Die vollen Namen aller Elemente? Betrag.anzahlOb, Betrag.getAnzahlOb, Betrag.main betrag1.betrag, betrag1.getBetrag, betrag1.setBetrag betrag2.betrag, betrag2.getBetrag, betrag2.setBetrag	
1.6. Ausgabe des Programms Betrag? -15 ist negativ! Anzahl Objekte: 2 Betrag1: 250	

Lösung 4:

```

1  static int anzahl(char c, String[] r) {
2
3      // Sonderfall "r ist eine null-Referenz" behandeln:
4      if (r == null) return 0;
5
6      // Alle anderen Faelle behandeln:
7      int erg = 0;
8      for (int ri=0; ri<r.length; ri++) {          // ri: Index fuer Reihung
9          String s = r[ri];
10         for (int si=0; si< s.length(); si++) {    // si: Index fuer String
11             if (s.charAt(si) == c) erg++;
12         } // for
13     } // for
14
15     return erg;
16 } // anzahl

```

Lösung 5:

```

1  static boolean sindDisjunkt(double anfang1, double laenge1,
2                             double anfang2, double laenge2) {
3      return (anfang1+laenge1 < anfang2) ||
4             (anfang2+laenge2 < anfang1);
5  } // sindDisjunkt

```