

Vorname

Nachname

Matrikel-Nr

Tragen Sie Ihre Lösungen zu den **Aufgaben 1 bis 4** in das folgende **Formular** ein. Schreiben Sie Ihre Lösung zu **Aufgabe 5** auf ein extra Blatt (oder mehrere Blätter) und kennzeichnen Sie dieses Blatt (diese Blätter) oben links mit Ihrem **Nachnamen**.

**Formular** für die **Lösungen** zu den Aufgaben 1 bis 4:

1.1.	3.1. (erg1)
1.2.	3.2. (erg2)
1.3.	3.3. (erg3)
1.4.	3.4.
	3.5.
1.5.	3.6.
	3.7.
2.1.	3.8.
2.2.	3.9.
2.3.	3.10.
2.4.	3.11.
4.1. (Module)	
4.2. (int-Variablen)	
4.3. (Summe aller int-Variablen)	

**Aufgabe 1** (20 Punkte): Was wird zum Bildschirm ausgegeben? Geben Sie für jede Befehlsfolge die **Zeile** (bzw. **Zeilen**) an, die zum Bildschirm ausgegeben wird (bzw. ausgegeben werden):

```
1 // 1.1. -----
2 StringBuffer sb1 = new StringBuffer("ABCDEF");
3 for (int i=0; i<sb1.length(); i++) {
4     sb1.setCharAt(i++, 'X');
5 }
6 System.out.println("sb1 : " + sb1);

1 // 1.2. -----
2 StringBuffer sb2 = new StringBuffer("ABCDEF");
3 for (int i=sb2.length(); i>0; i--) {
4     sb2.setCharAt(--i, 'Y');
5 }
6 System.out.println("sb2 : " + sb2);

1 // 1.3. -----
2 int[] ir1 = {5, 3, 9, 6, 8};
3 int sum = 0, i = 0;
4 while (sum < 20) sum += ir1[i++];
5 System.out.println("sum : " + sum);

1 // 1.4. -----
2 for (int j=3; j>=0; j--) {
3     int ein = j;
4     do {
5         System.out.print(ein % 2);
6         ein /= 2;
7     } while (ein > 0);
8     System.out.println();
9 }

1 // 1.5. -----
2 for (int j=0; j<=4; j++) {
3     switch (j) {
4         case 3: System.out.print("eins ");
5         case 2: System.out.print("plus eins ");
6         case 1: System.out.print("plus eins "); break;
7         case 0: System.out.print("null"); break;
8         default: System.out.print("Hallo!"); break;
9     }
10    System.out.println();
11 }
```

**Aufgabe 2** (15 Punkte): Betrachten Sie die folgende Klasse Chloe:

```
1 public class Chloe {
2
3     public static int liesInt(int min, int max) throws java.io.IOException,
4                               Throwable {
5         int erg = Lesen.liesInt();
6         if (erg < min) throw new ArithmeticException("'" + erg);
7         if (max < erg) throw new Throwable      ("'" + erg);
8         return erg;
9     } // liesInt
10
11    public static void main(String[] s) {
12        int ein;
13        while (true) {
14            System.out.print("Eine Ganzzahl (100..200) oder q zum Beenden: ");
15            try {
16                ein = liesInt(100, 200);
17                System.out.println("Sie haben " + ein + " eingegeben!");
18            }
19            catch (ArithmeticException aus) {
20                System.out.println(aus.getMessage() + " ist zu klein!");
21            }
22        }
23    }
24 }
```

```

22         catch (NumberFormatException aus) {
23             if (aus.getMessage().equals("q")) break;
24             System.out.println("Falsche Eingabe " + aus.getMessage());
25         }
26         catch (Throwable aus) {
27             System.out.println(aus.getMessage() + " ist falsch!");
28         }
29     } // while
30 } // main
31 } // class Chloe

```

Was gibt dieses Programm **Chloe** aus, wenn der Benutzer die folgenden Daten eingibt (für den Lesebefehl in Zeile 16):

- 2.1. **150**
- 2.2. **50**
- 2.3. **250**
- 2.4. **abc**

**Aufgabe 3** (15 Punkte): Betrachten Sie die folgenden vier Klassen Carl, Carola, Caroline und Test::

```

1 class Carl {
2     static int anz = 0;
3     static int getAnz() {return anz;}
4     int n = 17;
5     void incrN() {n++; anz++;}
6 }

7 class Carola extends Carl {
8     int add(int a) {n += a; return n;}
9 }

10 class Caroline extends Carola {
11     int sub(int a) {n -= a; return n;}
12 }

13 class Test {
14     public static void main(String[] s) {
15         Carl c1 = new Carl();
16         Carola c2 = new Carola();
17         Caroline c3 = new Caroline();
18         c1.incrN();
19         c2.incrN();
20         c3.incrN();
21         int erg1 = Carl.getAnz();
22         int erg2 = c3.add(5);
23         int erg3 = c3.sub(7);
24         ...
25         // -----
26     } // main
27 } // class Test

```

- 3.1. Mit welchem Wert wird die Variable **erg1** (in Zeile 21) initialisiert?
- 3.2. Mit welchem Wert wird die Variable **erg2** (in Zeile 22) initialisiert?
- 3.3. Mit welchem Wert wird die Variable **erg3** (in Zeile 23) initialisiert?

Welche der folgenden Sätze sind **wahr** und welche sind **falsch**?

- 3.4. c3 ist ein Caroline-Objekt.
- 3.5. c3 ist ein Carl-Objekt.
- 3.6. c2 ist ein Caroline-Objekt.
- 3.7. c2 ist ein Carl-Objekt.
- 3.8. Carl ist die direkte Oberklasse von Caroline.
- 3.9. Carl ist eine Oberklasse von Caroline.
- 3.10. Caroline ist eine Unterklasse von Carl.
- 3.11. Carola ist eine Unterklasse von Caroline.

**Aufgabe 4** (10 Punkte): Betrachten Sie die folgenden Klassenvereinbarungen:

```
1 class AKlasse {
2     static int otto = 1;
3     int emil = 100;
4 }
5
6 class BKlasse extends AKlasse {
7     static int anna = 50;
8     int berta = 150;
9 }
10
11 public class TestAB {
12     public static void main(String[] s) {
13         AKlasse a1;
14         AKlasse a2 = new AKlasse();
15         BKlasse b1 = new BKlasse();
16         ...
17     } // main
18 } // class TestAB
```

Führen Sie das Programm **TestAB** aus ("im Kopf" oder "mit Papier und Bleistift") bis Sie die **Zeile 16** erreichen. Beantworten Sie für diesen Moment die folgenden Fragen:

- 4.1. **Wieviele Module** gehören in diesem Moment zum Programm TestAB und **wie heissen sie** (die Module)?
- 4.2. **Wieviele int-Variablen** existieren in diesem Moment und **wie heissen sie** (die int-Variablen)?  
Geben Sie die **vollen Namen** der Variablen an (z.B. **Moses.verena** für eine Variable namens **verena**, die in einem Modul namens **Moses** "lebt").
- 4.3. Welche Zahl ergibt sich, wenn man die Werte all dieser int-Variablen **addiert**?

**Aufgabe 5** (30 Punkte): Angenommen, Sie haben mehrere Mengen von Ganzzahlen, und zwar in Form von **int-Reihungen**. Hier ein paar **Beispiele** für solche Mengen, die als int-Reihungen dargestellt wurden:

```
int[] melissa    = {+17, +23, -18, +39, -55};
int[] melisande  = {-18, +35, +17, -19, +23, +55};
int[] melwina    = {+11, +22, +33, +44, +55};
int[] merita     = {};
int[] merle      = null;
... etc.
```

Sie können sich darauf verlassen, dass keine der Reihungen eine Zahl **mehr als einmal** enthält (weil in einer Menge kein Element mehrmals vorkommen darf). Die Beispielreihungen **merita** und **merle** sollen aber deutlich machen, dass auch gewisse extreme int-Reihungen vorkommen können (**leere Reihungen** und **Null-Referenzen**). Eine Reihung wie **merle** (eine Null-Referenz) soll als Darstellung einer **leeren Menge** behandelt werden.

Sie sollen eine Methode namens **schnittMenge** programmieren, mit der man den Durchschnitt von zwei solchen Mengen, berechnen kann. Hier ein "Skelett" dieser Methode:

```
public int[] schnittMenge(int[] ir1, int[] ir2) {
    // Verlaesst sich darauf, dass ir1 und ir2 Mengen darstellen ("keine doppelten
    // Elemente"). Liefert die Schnittmenge der Mengen ir1 und ir2.
    // Eine Null-Referenz wird als Darstellung einer leeren Menge behandelt.
    ...
}
```

Und hier ein paar Beispiele für **Aufrufe** dieser Methode:

```
int[] erika      = schnittMenge(melissa, melisande);
int[] erda       = schnittMenge(melwina, melissa);
int[] ermina     = schnittMenge(melissa, merita);
int[] erwine     = schnittMenge(melissa, merle);
... etc.
```

Die Methode **schnittMenge** soll bewirken, dass die Menge (bzw. Reihung) **erika** genau drei Elemente enthält, und zwar +17, +23 und -18. Die Menge **erda** muss leer sein (weil die Mengen **melwina** und **melissa** keine gemeinsamen Elemente haben). Die Menge **ermina** muss ebenfalls leer sein, weil **merita** leer ist und **erwine** muss leer sein, weil **merle** als Darstellung einer leeren Reihung behandelt werden soll.

**Hinweis:** Achten Sie darauf, dass das **Ergebnis** Ihrer Methode eine Reihung "der richtigen Länge" ist. Z.B. soll die Länge der Reihung **erika** gleich **3** sein (und nicht grösser).

Lösungen zu den **Aufgaben 1 bis 4** (eingetragen in das **Formular** für Lösungen):

1.1. <b>sb1 : XBXXDF</b>	3.1. (erg1) Mit dem Wert <b>3</b>
1.2. <b>sb2 : AYCXY</b>	3.2. (erg2) Mit dem Wert <b>23</b>
1.3. <b>sum : 23</b>	3.3. (erg3) Mit dem Wert <b>16</b>
1.4. <b>11</b>	3.4. <b>wahr</b>
<b>01</b>	3.5. <b>wahr</b>
<b>1</b>	
<b>0</b>	
1.5. <b>null</b>	3.6. <b>falsch</b>
<b>plus eins</b>	
<b>plus eins plus eins</b>	3.7. <b>wahr</b>
<b>eins plus eins plus eins</b>	
<b>Hallo!</b>	
2.1. <b>Eine Ganzzahl ... : 150</b>	3.8. <b>falsch</b>
<b>Sie haben 150 eingegeben!</b>	
2.2. <b>Eine Ganzzahl ... : 50</b>	3.9. <b>wahr</b>
<b>50 ist zu klein!</b>	
2.3. <b>Eine Ganzzahl ... : 250</b>	3.10. <b>wahr</b>
<b>250 ist falsch!</b>	
2.4. <b>Eine Ganzzahl ... : abc</b>	3.11. <b>falsch</b>
<b>Falsche Eingabe abc</b>	
4.1. (Module) <b>5</b> Module gehören in diesem Moment zum Programm, sie heissen <b>TestAB, AKlasse, BKlasse, a2, b1</b>	
4.2. (int-Variablen) <b>5</b> int-Variablen existieren in diesem Moment, sie heissen <b>AKlasse.otto, BKlasse.anna, a2.emil, b1.emil, b1.bertha</b>	
4.3. (Summe aller int-Variablen) Die Summe aller int-Variablen ist gleich <b>401</b>	

**Lösung 5:** Die Methode **schnittMenge**:

```
19 public static int[] schnittMenge(int[] ir1, int[] ir2) {
20     // Verlaesst sich darauf, dass ir1 und ir2 Mengen darstellen.
21     // Liefert den Durchschnitt der Mengen ir1 und ir2. Eine Null-
22     // Referenz wird als Darstellung einer leeren Menge behandelt.
23
24     int[] erg0 = {};
25     if (ir1 == null || ir1.length == 0 ||
26         ir2 == null || ir2.length == 0) return erg0;
27
28     int laenge = Math.min(ir1.length, ir2.length);
29     int[] erg1 = new int[laenge];
30     int efi = 0; // erster freier Index von erg
31
32     for (int i1=0; i1<ir1.length; i1++) {
33         for (int i2=0; i2<ir2.length; i2++) {
34             if (ir1[i1] == ir2[i2]) erg1[efi++] = ir1[i1];
35         }
36     }
37     int[] erg2 = new int[efi];
38     for (int i=0; i<erg2.length; i++) erg2[i] = erg1[i];
39
40     return erg2;
41 }
```