

Vorname Nachname Matrikel-Nr

Schreiben Sie jede **Lösung** möglichst auf die Vorderseite eines **neuen** Blattes (und lassen Sie die Rückseiten Ihrer Lösungsblätter **leer**).

**Aufgabe 1** (20 Punkte): Schreiben Sie ein Unterprogramm entsprechend der folgenden Spezifikation:

```
1 static boolean maxDiff1(String s1, String s2) {
2     // Liefert true, wenn s1 und s2 gleich lang sind und hoechstens
3     // an einer Indexposition verschiedene Zeichen (ansonsten aber
4     // an allen Indexpositionen gleiche Zeichen) haben. Liefert sonst
5     // false. Beispiele
6     //
7     // maxDiff1("ACAC", "ACTC") ist gleich true
8     // maxDiff1("ACAC", "AAAC") ist gleich true
9     // maxDiff1("ACAC", "ACCA") ist gleich false
10    // maxDiff1("ACAC", "ACAC") ist gleich true
11    // maxDiff1("ACAC", "ACACA") ist gleich false
12    // maxDiff1("", "") ist gleich true
13    // maxDiff1("A", "") ist gleich false
```

**Aufgabe 2** (20 Punkte): Schreiben Sie ein Unterprogramm entsprechend der folgenden Spezifikation:

```
1 static int[] histo(String s) {
2     // Zaehlt, wie oft in s die Buchstaben 'A', 'C', 'G', 'T' und
3     // andere Zeichen vorkommen. Liefert eine int-Reihung erg der
4     // Laenge 5, fuer deren Komponenten gilt:
5     // erg[0] enthaelt die Anzahl der 'A's
6     // erg[1] enthaelt die Anzahl der 'C's
7     // erg[2] enthaelt die Anzahl der 'G's
8     // erg[3] enthaelt die Anzahl der 'T's
9     // erg[4] enthaelt die Anzahl der anderen Zeichen.
10    // Beispiel:
11    // Der Aufruf histo("AACXXGTGTGY?") liefert die Reihung
12    // {2, 1, 3, 3, 4}
```

**Aufgabe 3** (15 Punkte): Betrachten Sie die folgenden Klassenvereinbarungen:

```
1 class Karl {
2     public Karola next1;
3     public Karola next2;
4     public int data;
5
6     public Karl(Karola next1, Karola next2, int data) {
7         this.next1 = next1;
8         this.next2 = next2;
9         this.data = data;
10    } // Konstruktor Karl
11 } // class Karl
12 // -----
13 class Karola {
14     public Karl next;
15     public String data;
16
17     public Karola(Karl next, String data) {
18         this.next = next;
19         this.data = data;
20     } // Konstruktor Karola
21 } // class Karola
22 // -----
23 class Marlies {
24     static public double data = 2.5;
25
26     static public void main(String[] susi) {
27         Karola k01 = new Karola(null, "Apfel");
28         Karola k02 = new Karola(null, "Birne");
29         Karl k03 = new Karl(k01, k02, 17);
30         System.out.println("k03.next2.data: " + k03.next2.data);
31         ...
32     } // main
33
34 } // class Marlies
```

Führen Sie das Programm **Marlies** mit Papier und Bleistift aus. Wenn Sie dabei neue Referenzen erfinden müssen, dann verwenden Sie möglichst die Zahlen [1], [2], [3], ... etc. (in dieser Reihenfolge). Wie sehen die Variablen **k01**, **k02** und **k03** in dem Moment aus, in dem der Ausführer zur **Zeile 30** kommt? Zeichnen Sie die Variablen **k01**, **k02** und **k03** als Bojen.

**Tip:** Zeichnen Sie (zumindest erstmal) mit **Bleistift** und möglichst "**großzügig**".

**Aufgabe 4** (15 Punkte): Betrachten Sie noch einmal das Programm **Marlies** aus der vorigen Aufgabe und stellen Sie sich noch einmal den Moment vor, in dem der Ausführer die **Zeile 30** erreicht. Was gilt in diesem Moment? Beantworten Sie die folgenden Fragen entsprechend:

- 4.1. Wieviele **Klassen** hat der Ausführer bis dahin geladen und wie heißen diese Klassen?
- 4.2. Wieviele **Objekte** hat der Ausführer bis dahin erzeugt? Geben Sie für jedes dieser Objekte einen **Namen** an.
- 4.3. Wieviele **Module** gehören in diesem Moment zum Programm **K3**?
- 4.4. Geben Sie für jeden **Modul** an, **wieviele Konstruktoren**, **Attribute** und **Methoden** er enthält, und wie die Attribute und Methoden **heißen**, etwa so:  
"Der Modul **otto** enthält **2** Konstruktoren, **3** Attribute namens **x**, **y** und **summe** und **2** Methoden namens **berechneSumme** und **halbiere**".
- 4.5. Was gibt der Ausführer zum Bildschirm aus wenn er den Befehl in Zeile 30 ausführt?

**Aufgabe 5** (15 Punkte): Betrachten Sie das folgende Java-Programm:

```

1 class Aufgabe5 {
2     static class AusA extends Exception {};
3     static class AusB extends AusA {};
4     static class AusC extends AusB {};
5
6     static void pln(Object ob) {System.out.println(ob);}
7     // -----
8     static public void main(String[] susi) {
9         for (int n=1; n<=3; n++) {
10             try {
11                 upro(n);
12             } catch (AusC aus) {
13                 pln("A n: " + n);
14             } catch (AusB aus) {
15                 pln("B n: " + n);
16                 System.exit(1);
17             } catch (Throwable a) {
18                 pln("C n: " + n);
19             } finally {
20                 pln("D n: " + n);
21             } // try/catch/finally
22             pln("E n: " + n);
23             pln("-----");
24         }
25     } // main
26     // -----
27     static public void upro(int n) throws AusA {
28         try {
29             switch(n) {
30                 case 1: throw new AusA();
31                 case 2: throw new AusC();
32                 default: throw new AusB();
33             } // switch
34         } catch (AusB aus) {
35             pln("F n: " + n);
36             throw aus;
37         } // try/catch
38     } // upro
39     // -----
40 } // class Aufgabe5

```

Geben Sie an, was dieses Programm **zur Standardausgabe** (d.h. zum Bildschirm) **ausgibt**.

**Aufgabe 6** (15 Punkte): Beantworten Sie die folgenden Fragen möglichst kurz aber genau:

6.1. Was ist ein Modul?

6.2. Was ist eine Klasse?

6.3. Die in einer Klasse vereinbarten **Elemente** haben wir auf drei verschiedene Weisen (nach Art, nach Sichtbarkeit und nach "Aspektzugehörigkeit") **in Gruppen eingeteilt**. Geben Sie für jede der drei Weisen die Gruppen an.

6.4. Angenommen, Sie haben folgende Klasse:

```

1 class K1 {
2     ...
3     public int otto(int n) {...}
4     ...
5 } // class K1

```

Was müsste der Programmierer machen, damit der Name otto **überladen** wird?

6.5. Angenommen, Sie haben folgende Klasse:

```

6 class K2 {
7     ...
8     public int emil(int n) {...}
9     ...
10 } // class K2

```

Was müsste der Programmierer machen, damit die in Zeile 8 vereinbarte Methode (d.h. die Methode mit der Signatur **emil int**) **überschrieben** wird?