!!! Die Liste der Übungen erhebt keinen Anspruch auf Vollständigkeit. !!!

Inhaltsverzeichnis

1. Grundlagen, Variablen

1.1. Variablennamen. (M:36)

Welche der folgenden Symbole sind gültige Variablennamen in Java?

maxvalue	end	10%ofSum
maxValue	End	10procOfSum
max_value	int	sum10
max value	myInt	_10PercentOfSum

1.2. Deklaration und Initialisierung.

Was heißt, eine Variable zu deklarieren? Welchen Wert hat die Variable nach der Deklaration? Was ist der Unterschied zwischen Deklaration und Initialisierung? Welchen Wert hat die Variable nach der Initialisierung?

1.3. Boolesche Ausdrücke. (M:51)

Vereinfachen Sie die folgenden booleschen Ausdrücke nach den Regeln von DeMorgan.
Die Regeln von Augustus DeMorgan:

```
!(a && b) ≡ !a || !b
!(a || b) ≡ !a && !b
```

Ausdrücke:

```
!(x < y && y < z)
(x != y) || !(y == z && y == x)
!(x >= -3 && x <= 0) && 5 < x
```

1.4. Boolesche Ausdrücke (2). (M:51)

Was sind die Werte der Variablen a, b und c, wenn im folgenden Programmstück nacheinander für x die Eingabewerte -1, 0, 5 und 10 gelesen werden?

```
int x = In.readInt();
boolean a = x > 0 && x <= 10;
boolean b = x < 5 || x > 9;
boolean c = !(b || a);
```

1.5. Lebensdauer und Sichtbarkeit von Variablen. (M:89)

Geben Sie an, welche Variablen an den mit /* i */ gekennzeichneten Stellen im folgenden Programm leben und welche davon sichtbar sind.

```
class Program {
     static in t factor = 10;

     static int max(int a, int b) { /* 3 */
          if (a > b) return a;
          else return b;
     }

     static int compute(int x) { /* 2 */
          int y  = factor * max(x, 0);
          return y;
     }

     public static void main(String[] args) { /* 1*/
          int n = Integer.valueOf(args[0]);
          System.out.println(compute(n));
     }
}
```

2. Experimente

2.1. Dreieck. (M:51)

Schreiben Sie ein Programm, das die Seitenlängen eines Dreiecks einliest und prüft, ob es ein gleichseitiges, ein gleichschenkeliges, ein rechtwinkeliges, ein sonstiges gültiges oder ein ungültiges Dreieck ist. Ein Dreieck ist ungültig, wenn die Summe zweier Seitenlängen kleiner oder gleich der dritten Seitenlänge ist. Beachten Sie, dass ein Dreieck sowohl rechtwinkelig als auch gleichschenkelig sein kann.

2.2. Mittelwert.

Schreiben Sie ein Programm, das den mittleren Wert in einem int-Array zurückgibt.

2.3. Multiplikationstabelle. (M:20)

Schreiben Sie ein Programm, das für einen beliebigen Wert n eine Multiplikationstabelle der Größe n mal n ausgibt, in der das Element in Zeile i und Spalte j den Wert i*j hat.

2.4. Magisches Quadrat. (M:107)

Ein magisches Quadrat ist eine quadratische Matrix, für die die Summe jeder Zeile, jeder Spalte und der beiden Diagonalen denselben Wert ergibt. Lesen Sie eine Zahl n sowie eine n x n-Matrix ein und stellen Sie fest, ob es sich bei ihr um ein magisches Quadrat handelt.

3. Methoden

3.1. Fehlern ausweichen.

Was kann/soll man hier alles absichern und welche Möglichkeiten gibt es dafür?

```
public void setWord(int i, Word w){
     s[i]=w;
}

public ArrayList<String> leseDatei(String fileName) {
     BufferedReader br = new BufferedReader(new InputStreamReader(new FileInputStream(filename), "UTF-8"));
```

```
     String line = null;

     for (line = br.readLine(); line != null; line = br.readLine()) {
          ...
     }
}

public static void main(String[] args) {
     sum(args[0], args[1]);
}
```

3.2. Operator ++ (R:81)

Was ist die Ausgabe?

```
public class Plus {
     public static void main (String args []) {
          int a = 1, b = 2, c = 3, d = 4;
          System.out.println(++a);
          System.out.println(a);
          System.out.println(b++);
          System.out.println(b);
          System.out.println((++c) + (++c));
          System.out.println(c);
          System.out.println((d++) + (d++));
          System.out.println(d);
     }
}
```

3.3. Finde den Fehler. (R:97)

Diese Klasse enthält vier (logische) Fehler.

```
public class Falsch {
     public static void main (String[] args) {
          int x = 0, y = 4;

          // Beispiel A
          if (x < 5)
```

```
            if (x < 0)
                 System.out.println("x < 0");
        else
            System.out.println("x >= 5");

        // Beispiel B
        if (x > 0)
            System.out.println("ok! x > 0");
            System.out.println("1/x = " + (1/x));

        // Beispiel C
        if (x > 0);
            System.out.println("1/x = " + (1/x));

        // Beispiel D
        if (y > x) {  // vertausche x und y
            x = y;
            y = x;
        }
        System.out.println("x = " + x + "     y = " + y);
    }
}
```

3.4. Finde den Fehler (2). (R:180)

Die nachfolgenden Programmfragmente weisen jeweils einen syntaktischen, bzw. semantischen Fehler auf.

```
//1.
public void quadrat (double x) {
    reutrn Math.pow(x, 2);
}
//2.
double[][] matrix3x3 = {1.0, 2.0, 3.0,
                        4.0, 5.0, 6.0,
                        7.0, 8.0, 9.0};
//3.
double out = 3.1e5, println = 0.5;
system.out.println(out+println);
```

3.5. Initialisierung.

Welche Form ist richtig?

```
HashMap<String, Integer> map = new HashMap<String, Integer>();
Map<String, int[]> map = new Map<String, int[]>();
Map<String, int[]> map = new HashMap<String, int[]>();
HashMap<String, String> map = new Map<String, String>();
HashMap<String, String> map = new Object<String, String>();
HashMap<String, String> map = new Object();
Object map = new HashMap<Double, Double>();
```

3.6. Rekursion.

Welche Ausgabe erzeugt folgende Methode, wenn man sie mit p(4) aufruft?

```
public static void p(int x) {
     if (x > 0) {
          p(x - 1);
     }
     System.out.println(x);
}
```

4. Arrays

4.1. Array-Erzeugung.

Auf wie viele Weise kann man ein String-Array wie dieses erzeugen? (Implementiere mindestens drei Möglichkeiten.)

```
["a", "b", "c", "d", "e"]
```

4.2. Kehre die Elemente eines Arrays "in place" um.

4.3. Implementiere einen Sortieralgorithmus deiner Wahl.

(Bubble-Sort: Wenn zwei benachbarte Elemente in der falschen Sortierreihenfolge stehen, vertauscht man sie. Wenn im gesamten Array keine Vertauschungen mehr möglich sind, ist das Array sortiert.)

5. Klassen

5.1. instanceof und getClass

Was ist die Ausgabe?

```
Circle c = new Circle(3.5);
SubCircle sc = new SubCircle(3.5);

System.out.println("c instanceof Circle:\t" + (c instanceof Circle));
System.out.println("c instanceof SubCircle:\t" + (c instanceof SubCircle));
System.out.println("sc instanceof Circle:\t" + (sc instanceof Circle));
System.out.println("sc instanceof SubCircle:\t" + (sc instanceof SubCircle));
System.out.println("c.getClass():\t" + c.getClass());
System.out.println("sc.getClass().getSimpleName():\t" + sc.getClass().getSimpleName());
System.out.println("c.getClass() == sc.getClass():\t" + (c.getClass() == sc.getClass()));

Circle c2 = new SubCircle(3.5);
SubCircle sc2 = (SubCircle)c2;
SubCircle sc3 = null;

System.out.println("sc3 instanceof Circle:\t" + (sc3 instanceof Circle));
System.out.println("sc3 instanceof SubCircle:\t" + (sc3 instanceof SubCircle));
System.out.println("sc3.getClass().getSimpleName():\t" + sc3.getClass().getSimpleName());
```

5.2. Import

Wie könntest du erreichen, dass PI und pow() im folgenden Programm ohne die Klassenangabe verwendet werden können?

```
package prog2.shapes;
import static prog2.util.Colored.Color;

public class Circle extends ColoredShape {
     private double radius;
     ...
     @Override
     public double area() { return Math.PI * Math.pow( radius, 2 ); }
     ...
}
```

5.3. Welche der sechs Klassen ist richtig? (R: Aufgabe 8.7)

```
public class Fehler1 {
     private String name;

     public Fehler1(String name) {
          name = name;
     }

     @Override
     public String toString() {
          return "Name = " + name;
     }

     public static void main(String[] args) {
          System.out.println(new Fehler1("Testname"));
     }
}

public class Fehler2 {
     private String name;

     public Fehler2(String name) {
          this.name = name;
     }
```

```
     @Override
     public String toString() {
          return "Name = " + name;
     }

     public static void main(String[] args) {
          System.out.println(new Fehler2("Testname"));
     }
}

public class Fehler3 {
     private String name;

     public Fehler3(String nom) {
          name = nom;
     }

     @Override
     public String toString() {
          return "Name = " + name;
     }

     public static void main(String[] args) {
          System.out.println(new Fehler2("Testname"));
     }
}

public class Fehler4 {
     private String name;

     public Fehler4(String nom) {
          name = nom;
     }

     @Override
     public String toString() {
          return "Name = " + name;
     }
```

```
     public static void main(String[] args) {
          System.out.println(new Fehler4("Testname"));
     }
}

public class Fehler5 {
     private String name;

     public void Fehler5(String name) {
          this.name = name;
     }

     @Override
     public String toString() {
          return "Name = " + name;
     }

     public static void main(String[] args) {
          System.out.println(new Fehler5("Testname"));
     }
}

public class Fehler6 {
     private String name;

     public void Fehler6(String nom) {
          name = nom;
     }

     @Override
     public String toString() {
          return "Name = " + name;
     }

     public static void main(String[] args) {
          Fehler6 variable = new Fehler6();
          variable.name = "Testname";
          System.out.println(variable);
     }
```

```
}
```

5.4. Vererbung – Konstruktoren

Was ist die Ausgabe?

```
public class Super {
     public String x = "vor Super-Konstruktor";

     public Super() {
          System.out.println("Super-Konstruktor gestartet.");
          System.out.println("x = " + x);
          x = "nach Super-Konstruktor";
          System.out.println("Super-Konstruktor beendet.");
          System.out.println("x = " + x);
     }
}

public class Sub extends Super {
     public String y = "vor Sub-Konstruktor";

     public Sub() {
          System.out.println("Sub-Konstruktor gestartet.");
          System.out.println("x = " + x);
          System.out.println("y = " + y);
          x = "nach Sub-Konstruktor";
          y = "nach Sub-Konstruktor";
          System.out.println("Sub-Konstruktor beendet.");
          System.out.println("x = " + x);
          System.out.println("y = " + y);
     }
}

public class SubSuperTest {
     public static void main(String[] args) {
          Sub sub = new Sub();
     }
}
```

5.5. Wie könnten die folgenden Variablen und Methoden implementiert werden?

```
public class Mensch {
     //Anzahl Menschen auf der Erde

     //Name

     //Geschlecht

     //Geburtsdatum

     //Kinder

     //seine/ihre Lieblingsbücher

     //istLebendig

     //lebt er/sie noch?

     //nBuch in die Bücherliste einfügen

     //clone (?) – wie verhindert man das?

     //Mensch-Konstruktor (oder andere Möglichkeiten zum Erzeugen eines Mensch-Objektes)
}
```

5.6. Prioritätenschlange. (M:164)

Implementieren Sie eine Klasse PriorityQueue, die Elemente nach Prioritäten verwaltet. Jedes Element hat zusätzlich zu seinen Daten eine Priorität zwischen 0 und 9. Die Operation put(x) fügt das Element x gemäß seiner Priorität in die Schlange ein. Die Operation get() liefert aus der Schlange das Element mit der höchsten Priorität. Enthält die Schlange mehrere Elemente gleicher Priorität, so sollen sie in der Reihenfolge geliefert werden, in der sie in die Schlange gestellt wurden.

5.7. Bibliotheksverwaltung. (M:179)

Implementieren Sie eine Bibliotheksverwaltung mit folgender Funktionalität:

- Eingeben von Büchern: Ein Buch hat einen Autor, einen Titel, eine ISBN-Nummer und einen Verlag, wobei die ISBN-Nummer eindeutig ist.
- Suchen eines Buches mit einer bestimmten ISBN-Nummer.
- Löschen eines Buches mit einer bestimmten ISBN-Nummer.
- Iterieren über alle Bücher, d.h. eine Zugriffsfunktion, die bei jedem Aufruf das nächste Buch der Bibliothek liefert.

Die Bibliothek soll durch eine Klasse Library realisiert werden. Sie soll Objekte der Klasse Book verwalten.

5.8. Textzentrierung. (M:231)

Entwerfen Sie ein Programm, das einen Text von einer Datei liest und auf einem Bildschirm mit fester Zeilenbreite zentriert ausgibt. Eine Zeile ist maximal n Zeichen lang und soll so viele Wörter wie möglich enthalten. Der Text kann aber nur an Stellen unterbrochen werden, an denen Leerzeichen stehen. Die Zentrierung soll durch Einfügen von Leerzeichen am Anfang eine Zeile erfolgen.

6. Textverarbeitung

6.1. Text einlesen.

Ein Satz eines Textes sieht folgendermaßen aus. Die Sätze werden voneinander durch eine leere Zeile getrennt. Wie würdest du den Text daraus extrahieren? Wie würdest du nur die Verben extrahieren?

```
wsj/02/wsj_0294.mrg.5.16          Bankruptcy     -     -    NN           (S1(S(NP*                -               *                 *
wsj/02/wsj_0294.mrg.5.16             experts     -     -   NNS                 *)                -               *                 *
wsj/02/wsj_0294.mrg.5.16                said     -     -   VBD               (VP*                -               *                 *
wsj/02/wsj_0294.mrg.5.16                 the     -     -    DT         (SBAR(S(NP*                -               *                 *
wsj/02/wsj_0294.mrg.5.16                 law     -     -    NN                 *)                -               *                 *
wsj/02/wsj_0294.mrg.5.16                  is     -     -   AUX               (VP*                -               *                 *
wsj/02/wsj_0294.mrg.5.16                 n't     -     -    RB                  *                -               *                 *
wsj/02/wsj_0294.mrg.5.16               clear     -     -    JJ            (ADJP*                 -               *                 *
wsj/02/wsj_0294.mrg.5.16                  on     -     -    IN              (PP*                 -               *                 *
wsj/02/wsj_0294.mrg.5.16                 how     -     -   WRB      (SBAR(WHADVP*)               -     (ARGM-MNR*)       (ARGM-MNR*)
wsj/02/wsj_0294.mrg.5.16                such     -     -    JJ            (S(NP*                 -          (ARG0*            (Agent*
wsj/02/wsj_0294.mrg.5.16                  an     -     -    DT                *                  -               *                 *
wsj/02/wsj_0294.mrg.5.16         arbitration     -     -    NN                *                  -               *                 *
wsj/02/wsj_0294.mrg.5.16              ruling     -     -    NN                *)                 -              *)                *)
wsj/02/wsj_0294.mrg.5.16                 can     -     -    MD              (VP*                 -               *                 *
wsj/02/wsj_0294.mrg.5.16              affect  31.1    01    VB              (VP*            affect             (V*)              (V*)
wsj/02/wsj_0294.mrg.5.16                   a     -     -    DT           (NP(NP*                 -          (ARG1*       (Experiencer*
```

```
wsj/02/wsj_0294.mrg.5.16          company     -     -    NN                 *                  -                  *                  *
wsj/02/wsj_0294.mrg.5.16               's     -     -   POS                *)                 -                  *                  *
wsj/02/wsj_0294.mrg.5.16             case     -     -    NN        *))))))))))                 -                 *)                 *)
wsj/02/wsj_0294.mrg.5.16                .     -     -     .               *))                 -                  *                  *
```

6.2. Text extrahieren mit RE.

Einige Zeilen eines Textes sehen folgendermaßen aus. Wie würdest du die e1-e2-Paare extrahieren? In welcher Datenstruktur würdest du sie speichern? Was sind Vor- und Nachteile dieser Datenstruktur?

```
003 "Earplugs relieve the <e1>discomfort</e1> from <e2>traveling</e2> with a cold allergy or sinus condition."
WordNet(e1) = "discomfort%1:26:00::", WordNet(e2) = "traveling%1:04:00::", Cause-Effect(e2,e1) = "true", Query =
"discomfort from *"

004 "We continue to see progress toward a world free of the <e1>daily terror</e1> of <e2>antipersonnel land mines</e2>."
WordNet(e1) = "terror%1:12:00::", WordNet(e2) = "land_mine%1:06:00::", Cause-Effect(e2,e1) = "true", Query = "terror of
*"
Comment: "daily terror" -> "terror"; (b) "antipersonnel landmines" -> "land mines"

005 "The Global Warming <e1>segment</e1> starts off with two, cute <e2>anecdotes</e2> about flowers blooming early and
aardvarks moving North."
WordNet(e1) = "segment%1:06:00::" , WordNet(e2) = "anecdote%1:10:00::", Cause-Effect(e2,e1) = "false", Query = "* starts
with *"
```

6.3. Extrahieren mit RE:

Extrahiere die Werte der beiden Argumente mit einem regulären Ausdruck.

```
  <property name="src.dir" value="${basedir}/src"/>
  <property name="bin.dir" value="${basedir}/bin"/>
  <property name="main.class" value="prog2.ShapeTest"/>
  <property name="jar.file" value="shapes.jar"/>
```

6.4. Text einlesen, extrahieren, schreiben. (M:115)

Entfernen von Kommentaren. Schreiben Sie ein Programm, das eine Java-Quelldatei liest, daraus alle Kommentare entfernt und das Ergebnis auf eine neue Datei schreibt.

6.5. Lauflängenkodierung. (M:126)

Die Lauflängenkodierung ist eine Komprimierungstechnik, bei der jede Zeichenfolge, die aus mehr als 2 gleichen Zeichen besteht, durch das Zeichen und die Länge der Folge codiert wird. Die Eingabe "ABBCCCKKKKKKK" wird zum Beispiel zu "ABBC3K7".
a) Schreiben Sie eine Methode, die einen Buchstaben-String nach diesem Verfahren codiert.
b) Schreiben Sie eine Methode, die einen nach diesem Verfahren codierten String decodiert.

6.6. Häufigkeit von Zeichen. (M:116)

Schreiben Sie ein Programm, das einen Text liest und die Häufigkeit der darin orkommenden Zeichen berechnet. Geben Sie die Zeichenhäufigkeiten als Histogramm aus, also zum Beispiel:

```
a     ******
b     **
e     *********
...
```

6.7. Anagramme. (M:126)

: Zwei Wörter sind Anagramme, wenn sie aus denselben Buchstaben in beliebiger anderer Reihenfolge bestehen (z.B. "Maus" und "Saum"). Schreiben Sie eine Methode isAnagram(String s1, String s2), die prüft, ob die beiden Strings s1 und s2 Anagramme sind und das Ergebnis als boolean-Wert zurückgibt. Groß- und Kleinschreibung soll ignoriert werden.

Quellen:

- (M) Hanspeter Mössenböck (2005): Sprechen Sie Java?. Eine Einführung in das systematische Programmieren. 3. Aufl., dpunkt.
- (R) Dietmar Ratz, Jens Scheffler, Detlef Seese, Jan Wiesenberger (2011): Grundkurs Programmieren in Java. 6. Aufl., Hanser.