Übungskatalog

Einführung in die Programmierung

Giovanni Mahlknecht *

20. Oktober 2004

Zusammenfassung

Dieses Dokument ist eine Sammlung von Programmieraufgaben, welche in der Unterrichtseinheit „Einführung in die Programmierung mit Java“ an der Handelsoberschule „Franz Kafka“, Fachrichtung „Mercurio“ in der 11. Klasse Verwendung finden. Wenn Sie Übungen zu dieser Sammlung beitragen möchten, schicken Sie diese bitte als Textdatei mit dem Betreff „Übungskatalog“ an g.mahlknecht@gmx.net.

Inhaltsverzeichnis

*Bedanken möchte ich mich im Besonderen bei Hildegard Frei für die Korrektur und die beigetragenen Übungen. Ebenso bedanke ich mich bei Werner Frei und Stefan Ganterer für die Übungen, welche sie zur Verfügung gestellt haben. Ideen zu einigen Übungen stammen aus dem Internet. Leider kann ich nicht mehr alle Herkunftsquellen angeben und entschuldige mich hierfür im Voraus.

1 Einfache Programme

1.1 Rechner

Erstelle ein Programm, welches einen einfachen Taschenrechner simuliert. Der Benutzer gibt zwei Zahlen vom Datentyp **int**(Integer) ein. Am Bildschirm werden die Summe $z1 + z2$, die Differenz $z1 - z2$, das Produkt $z1 * z2$ und der Quotient $\frac{z1}{z2}$ der beiden Zahlen ausgegeben.

1.2 Flächenberechnung Quadrat

Erstelle ein Programm, welches die Fläche und den Umfang eines Quadrats berechnet. Der Benutzer gibt über Tastatur die Seitenlänge ein.

Beispiel:

```
Gib die Seitenlänge des Quadrates (in cm) an: 5
Der Umfang beträgt: 20 cm
Die Fläche beträgt: 25 qcm
```

1.3 Flächenberechnung Rechteck

Erstelle analog zur Übung 1.2 ein Programm, welches die Fläche eines Rechtecks berechnet.

1.4 Flächenberechnung Kreis

Erstelle ein Programm, welches die Fläche und den Umfang eines Kreises berechnet. Verwende für die Kreiszahl π die Konstante vom Datentyp **double** Math.PI. Achtung: Verwende einen Datentyp, der Zahlen mit Dezimalstellen unterstützt.

1.5 Volumen und Oberfläche eines Quaders

Erstelle ein Programm, welches das Volumen und die Oberfläche eines Quaders berechnet. Die drei Größen Breite, Höhe, Tiefe können unterschiedlich sein.

1.6 Was gibt dieses Programm aus?

```
1 public class Ausgabe1
2 {
3   public static void main (String args[])
4   {
5     int i;
6     i = 10;
7     System.out.println("i");
8   }
9 }
```

Antworten:

Antwort	j/n	Begründung
10		
i		
Fehler beim Kompilieren		
Fehler bei der Ausführung		

1.7 Kommentare

Welche der folgenden Programmzeilen sind gültige Kommentare, oder mit anderen Worten welche der folgenden Programmzeilen verursacht beim Kompilieren einen

Fehler?

Programmzeile	Kommentar j/n
irgendwas;	
irgendwas; //	
// irgendwas	
// irgendwas;	
/* irgendwas */	
*/ irgendwas */	
/ irgendwas /	
/* irgendwas */;	
// /* irgendwas */	

1.8 Bezeichner

Welches sind gültige Bezeichner für Variablen und Programme in Java?

Bezeichner	j/n	Begründung
Micky Maus		
Micky_Maus		
caesar-brutus		
cäsar_brutus		
0 Ahnung		
0_Ahnung		
_1234A		
abc1		
diesisteinbesonderslangerprogrammname		

1.9 Pythagoräischer Lehrsatz

Erstelle ein Programm welches, nach Eingabe der beiden Katheten eines rechtwinkligen Dreiecks a und b mit Hilfe des pythagoräischen Lehrsatzes ($c^2 = a^2 + b^2$) die Länge der Hypothenuse c berechnet.

Tipp: Berechnung der Quadratwurzel mit der Funktion Math.sqrt(**double** n). Um z. B. die Wurzel aus 5 zu berechnen und in der Variable ergebnis vom Datentyp **double** abzuspeichern, wird folgende Anweisung verwendet:

```
1 double ergebnis;
2 ergebnis = Math.sqrt(5);
```

1.10 Übungen zum Casting

Was geben die folgenden Programme am Bildschirm aus? Beantworte die Frage zuerst ohne das Programm am Rechner, begründe deine Antwort, verifiziere dann deine Antwort mit Hilfe des Programms am Rechner.

1.10.1 Ausgabe 2

```java
public class Ausgabe2
{
  public static void main (String args[])
  {
    int var1=100;
    int var2=3;
    int var3;
    var3 = var1 / var2;
    System.out.println(k);
  }
}
```

Antworten:

Antwort	j/n	Begründung
33		
33.3333333333333		
Fehler beim Kompilieren		
Fehler bei der Ausführung		

1.10.2 Ausgabe 3

```java
public class Ausgabe3
{
  public static void main (String args[])
  {
    int var1=100;
    int var2=3;
    double var3;

    var3 = vat1 / var2 ;
    System.out.println(var3);
  }
}
```

Antworten:

Antwort	j/n	Begründung
33		
33.3333333333333		
Fehler beim Kompilieren		
Fehler bei der Ausführung		

1.10.3 Ausgabe 4

```java
public class Ausgabe4
{
  public static void main (String args[])
  {
    int var1=1000;
    int var2=3;
    double var3;

    var3 = 1.0 * var1 / var2 ;
    System.out.println(var3);
  }
```

```
12 }
```

Antworten:

Antwort	j/n	Begründung
333.0		
333.333333333333		
Fehler beim Kompilieren		
Fehler bei der Ausführung		

1.10.4 Ausgabe 5

```
1  public class Ausgabe5
2  {
3    public static void main (String args[])
4    {
5      int i=100;
6      int j=3;
7      float k;
8
9      k = (float) i / j ;
10     System.out.println(k);
11   }
12 }
```

Antworten:

Antwort	j/n	Begründung
33.33333		
33.33332		
Fehler beim Kompilieren		
Fehler bei der Ausführung		

1.10.5 Ausgabe 6

```
1  public class Ausgabe6
2  {
3    public static void main (String args[])
4    {
5      float i=100.0;
6      float j=3.0;
7
8      System.out.println( i / j );
9    }
10 }
```

Antworten:

Antwort	j/n	Begründung
33.33333		
33.33332		
Fehler beim Kompilieren		
Fehler bei der Ausführung		

1.11 Umrechnung von Temperaturen

Erstelle ein Programm, welches eine Temperatur von Celsius nach Fahrenheit umrechnet.

Formel zur Umrechnung: $FahrenheitGrad = 32 + CelsiusGrad * \frac{9}{5}$.

Achtung: $\frac{9}{5}$ mit ganzzahligen Werten berechnet, ergibt den ganzzahligen Wert 1. (Das eigentliche Ergebnis 1.8 wird auf die nächstkleinere ganze Zahl abgerundet.) Verwende die Division zweier **double** Werte: 9.0 / 5.0 oder 9d / 5d.

1.12 Berechnung der Mehrwertsteuer

Erstelle ein Programm, welches nach Eingabe des Nettopreises, die Mehrwertsteuer (zur Zeit 20 %) und den Endpreis berechnet. Achtung: es könnten Rundungsfehler auftreten. Überprüfe, ob diese auftreten und versuche gegebenenfalls zu erklären, warum.

Erstelle folgende Programme zur Mehrwertsteuerberechnung:

1. mit festem Warenwert von 130€ und Mehrwertsteuersatz von 20%
2. mit vom Benutzer einzugebenden Warenwert und festem Mehrwertsteuersatz von 20%
3. Sowohl der Warenwert als auch der Mehrwertsteuersatz wird vom Benutzer eingegeben.

Die Ausgabe soll in folgender Form vorliegen:

```
Warenwert:            ............
MwSt Betrag:          ............
-----------------------------------
Warenwert inkl. MwSt: ............
```

1.13 Berechnung der MwSt II

Erweitere dein Programm 1.12 so, dass die Berechnung gleichzeitig 3 Waren berücksichtigt und zusätzlich zu den einzelnen Beträgen auch der Gesamtbetrag ausgegeben wird.

Die Ausgabe soll in folgender Form vorliegen:

```
Warenwert 1:          ............
MwSt 1:               ............
Warenwert 2:          ............
MwSt 2:               ............
Warenwert 3:          ............
MwSt 3:               ............
-----------------------------------
Warenwert Summe:      ............
MwSt Summe:           ............
-----------------------------------
Warenwert inkl. MwSt: ............
```

1.14 Euro in Lire

Erstelle ein Programm, welches nach Eingabe eines Betrags in € den Betrag in Lire berechnet. Wechselkurs 1€ = 1936,27 Lire.

Zusatz: achte auf korrekte Rundung.

1.15 Lire in Euro

Erstelle ein Programm, welches nach Eingabe eines Betrags in Lire den Betrag in € berechnet. Wechselkurs: 1€ = 1936,27 Lire.

Zusatz: achte auf korrekte Rundung.

1.16 Sekunden umrechnen

Erstelle ein Programm welches eine eingegebene Anzahl an Sekunden in der uns gewohnten Form Stunden:Minuten:Sekunden ausgibt.

Beispiel:

Sekunden 3723

Ergebnis 1:2:03

1.17 Busunternehmen Fahrpreis

Wenn eine Schulklasse eine Lehrfahrt unternimmt berechnet ein Busunternehmen den Fahrpreis wie folgt: Pro Person und Kilometer werden 0,05 € berechnet. Jeder 10. Schüler erhält eine Freifahrt. Die Begleitperson(en) bezahlt nur den halben Fahrpreis. Schreibe ein Programm, welches für n Teilnehmer die Gesamtfahrkosten und die durchschnittlichen Kosten für den einzelnen ermittelt.

1.18 Fehlersuche

Finde die syntaktischen Fehler.

```
1  // SummeVonZweiZahlen.java
2  // Berechnet die Summe zweier Zahlen und gibt diese am
3  // Bildschirm aus
4
5  class SummeVonZwei
6  {
7    static void main(String[] args)
8    {
9      int ersteZahl;
10     int zweiteZahl;
11     int summeDerZahlen;
12
13     ersteZahl = 10
14     weiteZahl = 11;
15     summeDerZahlen == ersteZahl + zweiteZahl;
16     System.out.println("Die Summe der Zahlen "+ersteZahl+
17                                    "und "+zweiteZahl+
18                                    "ist: "summeDerZahlen);
```

```
19      }
20  }
```

1.19 Blauer Dunst

Es soll ein Programm erstellt werden, das einem Raucher die Kosten für sein Laster vorrechnet. Dabei sollen vom Benutzer folgende Eingaben erwartet werden:

- derzeitiges Alter (in Jahren)
- Anzahl der Jahre, die der Raucher bereits raucht
- durchschnittliche Zahl der Zigaretten, die er pro Tag raucht
- Name der Zigarettenmarke
- Durchschnittlicher Preis je Packung Zigaretten
- Gehalt an Nikotin und Kondensat der vom Raucher gewählten Marke (in mg)

Der Computer errechnet folgende Angaben und gibt diese mit sinnvollen Meldungen am Bildschirm aus:

- Kosten für den bisherigen Zigarettenkonsum (ohne Zinsen)
- Kosten bis zum 65. Lebensjahr bei gleichbleibendem Verbrauch
- aufgenommene Menge an Nikotin und Kondensaten (in Gramm und Kilogramm)
- Verkürzung der Lebenserwartung (ca. 2 Minuten je Zigarette)

Zusätze

1. Eingabenkontrolle (Bedingungen): Alle Benutzereingaben sollen auf ihre Zulässigkeit überprüft werden. Bei falschen Eingaben soll keine Berechnung stattfinden.
2. Eingabenkontrolle 2 (do- while Schleife): Jede Benutzereingabe soll so lange wiederholt werden, bis korrekte Werte eingegeben werden.
3. Zinseszins (Schleife): Gesamtbetrag der Aufwendungen für Zigarettenrauch vom Beginn der Sucht bis zum 65. Lebensjahr einschließlich einer 2%igen Verzinsung pro Jahr (mit Zinseszinsen). Hinweis: Berechne die Tabakkosten für jeweils ein Jahr. Berechne die Zinsen jeweils am Ende des Jahres das angereifte Kapital und addiere die Raucherkosten des gerade abgelaufenen Jahres.

   Beispiel: jährliche Kosten: 500,-€ (entspricht ungefähr 10 Zigaretten je Tag)

   Ende Jahr 1: 0€ + kein Zins da im 1. Jahr + Kosten im Jahr 1 (500€)=500€

   Ende Jahr 2: 500€ + 5% Zinsen auf 500€ (=25€) + Kosten Jahr 2 (500€) = 1.025€

   Ende Jahr 3: 1.025+ 5% Zinsen auf 1025€ (=51,25€) + Kosten Jahr 3 (=500€) = 1576,25€

   Ende Jahr 4: 1576,25+ 5% Zinsen auf 1576,25€ (=78,81€) + Kosten Jahr 3 (=500€) = 2.155,06€

4. Berechnung wiederholen (do- while Schleife): Die Berechnung soll auf Wunsch des Benutzers wiederholt werden können („Willst du eine erneute Berechnung durchführen (ja / Ja)?“). Wenn du noch keine Zeichenketten (Strings) kennst, kannst du ja eine Zahleneingabe vom Benutzer verlangen. „Willst du eine erneute Berechnung durchführen (1 = ja)“. Der Benutzer soll ebenso entscheiden können, ob für die erneute Berechnung eine andere Zigarettenmarke verwendet wird, ob also eine erneute Eingabe von Marke, Preis und Nikotin/Kondensatgehalt erforderlich ist oder nicht. Bei der ersten Berechnung müssen diese Daten auf jeden Fall eingelesen werden!

2 Bedingungen

2.1 Volljährig?

Erstelle ein Programm, welches nach Eingabe des Alters einer Person am Bildschirm ausgibt, ob die Person volljährig ist oder nicht.

2.2 Anzahl Tage im Monat

Erstelle ein Programm, welches für die Eingabe eines Monats die Anzahl seiner Tage ausgibt.

2.3 Note in wörtliche Beurteilung umwandeln

Erstelle ein Programm, welches eine eingegebene Note (4 bis 10) in eine wörtliche Beurteilung (vier bis zehn) umformt. Zwischennoten müssen nicht, können aber berücksichtigt werden.

2.4 Kontrolle negativer Radius bei Kreisberechnung

Erstelle ein Programm, welches den Umfang und Flächeninhalt eines Kreises berechnet, wobei aber geprüft werden soll, ob die Aufgabe geometrisch sinnvoll ist, mit anderen Worten, ob der Wert des Radius positiv ist. Nur ein positiver Wert für den Radius r wird zur Berechnung zugelassen sonst wird eine Fehlermeldung ausgegeben. Vergleiche hierzu Aufgabe 1.4.

2.5 Ehrenurkunde

Die Teilnehmer einer Sportveranstaltung erhalten eine Ehrenurkunde, wenn sie mehr als 1000 Punkte erreicht haben (Fall A); sie erhalten eine Siegerurkunde bei 800 bis 999 Punkten (Fall B); eine geringere Punktzahl wird nicht mit einer Urkunde bewertet (Fall C).

Schreibe ein Programm, das jeder eingegebenen Punktzahl den richtigen Fall zuordnet und die zugehörige Urkunde mit Punktezahl und passendem Text am Bildschirm ausgibt.

2.6 Kosten Mobiltelefon

Ein Schüler telefoniert durchschnittlich 534 Minuten im Monat (130 Minuten davon zwischen 08:00 und 18:00 Uhr) und verschickt 22 SMS. Es stehen 3 Telefongesellschaften zur Auswahl:

Gesellschaft	A	B	C
monatliche Grundgebühr	10,-	0,-	0,-
Gespräche 8-18 Uhr ct je Minute	40 ct.	50 ct.	60 ct.
Gespräche 18-8 Uhr ct je Minute	40 ct.	15 ct.	60 ct. 30 min. je Monat gratis
ct je SMS	22 ct.	20 ct. 10 SMS je Monat gratis	17 ct.

Tabelle 1: Preismodelle Mobilfunkanbieter

Erstelle ein Programm welches die Kosten aller Anbieter berechnet. Die notwendigen Daten zu den Telefongesprächen (Dauer und Anzahl SMS) werden vom Benutzer eingegeben, die Daten der Telefongesellschaften (Kosten je Gesprächsminute, Kosten je SMS usw.) sind im Programm als Konstanten einzutragen.

Zusatz (Bedingungen): Das Programm soll auch den Namen des günstigsten Anbieters am Bildschirm ausgeben.

Zusatz (Denken!): Vergleiche die realen Preismodelle der Mobilfunkanbieter in Italien und versuche einen Algorithmus zum Kostenvergleich aufzustellen.

2.7 Teilerprobe

Dem Computer werden zwei natürliche Zahlen a und b eingegeben. Der Rechner soll prüfen, ob b ein Teiler von a ist und einen entsprechenden Text ausgeben. Verwende zur Berechnung des Rests den Modulo Operator %.

2.8 Gehalt eines Vertreters

Das Gehalt eines Vertreters für Automobile errechnet sich wie folgt: 1000,- € Grundgehalt und für jedes verkaufte Fahrzeug erhält er zusätzlich 100,- €. Wenn er mehr als 20 Autos verkauft, erhält er zusätzlich eine Leistungsprämie von 500,- €. Erstelle ein Programm, welches nach Angabe der verkauften Fahrzeuge das Gehalt des Vertreters berechnet.

2.9 Versicherung mit Selbstbeteilung

Bei einer Schadensversicherung hat der Versicherte bei Schäden über 100,- € 20 % der Schadenssumme, mindestens jedoch 100,- € selbst zu tragen. Erstelle ein Programm, das nach Eingabe der Schadenshöhe den Anteil des Versicherten und die Zahlung der Versicherung ausgibt.

2.10 Aussendienstprämie

Ein Unternehmen bezahlt ihre Verkäufer im Außendienst (Vertreter) nach folgender Vereinbarung: der monatliche Verdienst beträgt 20 % es monatlichen Verkaufsumsatzes. Übersteigt dieser Umsatz 10.000,- €, so erhält der Verkäufer zusätzlich noch 11,5 % des Betrages der 10.000,- € übersteigt. als Erfolgsprämie. Schreibe ein Programm, das den Verdienst und gegebenenfalls die Prämie nach Eingabe des Monatsumsatzes ausgibt.

2.11 Kindergeld

Erstelle ein Programm welches das Kindergeld einer Familie berechnet. Die Höhe des Kindergeldes ist von der Kinderzahl abhängig: für das erste Kind 150,- €, für das zweite Kind *zusätzlich* 170,- €, für das 3. und jedes weitere Kind *zusätzlich* 220,- €.

Beispiel:
3 Kinder: 150,- € + 170,- € + 220,- € = 540,- €
5 Kinder: 150,- € + 170,- € + 220,- € + 220,- € + 220,- € = 980,- €

2.12 Preis für CDR

Ein Händler berechnet den Verkaufspreis für CDR's folgendermaßen: weniger als 10 Stück kosten 1,- € je Stück, werden zwischen 10 und 50 Stück gekauft so kosten diese 0,95 € je Stück, bei 51 bis 100 Stück 0,90 € je Stück, ab 100 Stück 0,85 € je Stück. Erstelle ein Programm welches nach Eingabe der Stückzahl den Gesamtpreis berechnet.

2.13 Versandhandel und Verpackungskosten

Beim „XXX Versand“ gelten folgende Berechnungsmodalitäten für die Verpackung: Unabhängig davon, wie hoch das Porto oder die Kosten für eine aufwändige Verpackung sind, wird eine anteilige Pauschale berechnet. Sie beträgt 2,50 € bei einem Bestellwert, der 250,- € übersteigt; 3,- € bei einem Bestellwert zwischen 100,- € und 250,- € und 3,50 € bei einem Bestellwert unter 100,- €. Erstelle ein Struktogramm, welches nach Angabe des Bestellwertes die Verpackungspauschale berechnet und ausgibt.

2.14 Handwerker Arbeitsstunden

Ein Handwerker möchte ein Berechnungsprogramm für seine geleisteten Arbeitsstunden. Nach Eingabe der Anzahl der Arbeitsstunden wird der abzurechnende Betrag

errechnet und am Bildschirm ausgegeben. Der Stundenpreis ist fest und beträgt 22,50 €. Darüber hinaus verlangt er noch eine Fahrtkostenpauschale. Liegen die Stundenkosten über 1.000,- € verlangt er keine Fahrtkosten, liegen sie zwischen 500,- € und 1.000,- € so berechnet er 30,- €, bei weniger als 500,- € Stundenkosten verlangt er 50,- €.

Da sich der Handwerker oft vertippt, soll die Eingabe der Arbeitsstunden überprüft werden: die Anzahl der Arbeitsstunden kann nicht negativ sein. Die Berechnung und Ausgabe des Ergebnisses soll nur bei korrekter Eingabe (also positivem Wert) erfolgen!

2.15 Handwerker Arbeitsstunden mit variablem Stundenpreis

Verändere das Programm Handwerker Arbeitsstunden so, dass der Stundenpreis nicht fest ist, sondern von der Anzahl der geleisteten Arbeitsstunden abhängt. Für die ersten 10 Arbeitsstunden berechnet der Handwerker 25,- € für die nächsten 10 Arbeitsstunden 22,50 € und für alle weiteren Stunden 20,- €.

2.16 Zinsrechnung

Erstelle ein Programm zur Zinsrechnung. Der Benutzer soll auswählen können, ob die Zinsen, das Kapital oder der Prozentsatz berechnet wird.

Beispiel:

```
Wähle aus, was du berechnen möchtest:
1) Zinsen
2) Prozentsatz
3) Kapital
Ihre Wahl: 3
Zinsen: 100000
Prozentsatz: 5
Tage: 200
Das Kapital beträgt: ......
```

2.17 Chaotische Bedingungen 1

Was gibt das folgende Programm aus? Beantworte die Frage zuerst ohne das Programm am Rechner, begründe deine Antwort, verifiziere dann deine Antwort mit Hilfe des Programms am Rechner.

```
1  Was gibt
2  public class Bedingungen1
3  {
4    public static void main (String args[])
5  {        {int a=100,b=20;
6
7      if (a<b)
8                if (a-b>0)
9                        System.out.println("eins");
10         else
```

```
11                  System.out.println("zwei");
12 }
13
14 }
```

Antworten:

Antwort	j/n	Begründung
eins		
zwei		
Fehler beim Kompilieren		
Fehler bei der Ausführung		

2.18 Chaotische Bedingungen 2

Was gibt das folgende Programm aus? Beantworte die Frage zuerst ohne das Programm am Rechner, begründe deine Antwort, verifiziere dann deine Antwort mit Hilfe des Programms am Rechner.

```
1  public class Bedingungen2
2  {
3    public static void main (String args[])
4    {
5  int a=100,b=20;
6
7                   if (a<b)
8          {
9           if (a-b>0)
10         {
11          System.out.println("eins");
12         }
13        else
14         {
15          System.out.println("zwei");
16 }
17 }
18
19 }
```

Antworten:

Antwort	j/n	Begründung
eins		
zwei		
Fehler beim Kompilieren		
Fehler bei der Ausführung		

2.19 Dreiecksüberprüfung

Der Benutzer gibt die drei Seitenlängen a b c eines Dreiecks ein. Das Programm soll feststellen ob die eingegebenen Längen überhaupt ein Dreieck ergeben und falls ja, ob es sich um ein gleichseitiges, gleichschenkliges oder ungleichschenkliges Dreieck handelt. Zudem soll festgestellt werden, ob es sich bei dem Winkel α zwischen den Katethen a und b um einen stumpfen, spitzen oder rechten Winkel handelt.

Aus der Geometrie kennen wir folgende Regeln:

Bei einem Dreieck mit den Schenkellängen a, b, c ist die Summe von zwei Schenkellängen immer größer als die dritte Schenkellänge.

Die Summe der Winkel beträgt 180 °.

Ein rechter Winkel hat 90 °, ein spitzer Winkel weniger als 90 °, ein stumpfer Winkel mehr als 90 °. Aber das dürfte Dir ja bekannt sein. Oder etwa nicht?

Bei rechtwinkligen Dreiecken gilt der Pythagoräische Lehrsatz $c^2 = a^2 + b^2$.

2.20 Idealgewicht

Erstelle verschiedene Programme, welche feststellen, ob eine Person Über- Unter- oder Idealgewicht hat. Verwende dazu folgende Berechnungsformeln:

1. Methode 1: $Wert = \frac{groesse-100}{gewicht*100}$. Bei Frauen sollte das Ergebnis zwischen 88 und 98 liegen, bei Männern zwischen 93 und 97.
2. Methode 2: Bei Männern gilt als Idealgewicht in Kilogramm 95% von (Körpergröße -100), bei Frauen 90%. Es wird jeweils eine Toleranzschwelle von $\pm 5\%$ angewandt. Alles was darüber liegt, ist Übergewicht, darunter Untergewicht.
3. Der Body-Mass-Index wird mit der Formel
   $bmi = \frac{gewicht}{groesse^2}$
   berechnet. Das Gewicht wird in kg angegeben, die Größe in m. Mit Tabelle 2 kann der BMI interpretiert werden:

Gewichtsklasse	BMI
Untergewicht	< 18
Normalgewicht	18–25
leichtes Übergewicht	25–27
mässiges Übergewicht	27–30
starkes Übergewicht, Fettsucht (Adipositas)	>30

Tabelle 2: Interpretation BMI

2.21 Body Mass Index

Der Body Mass Index wird, wie bereits in Aufgabe 2.20 gesehen, mit folgender Formel berechnet: $bmi = \frac{gewicht}{groesse^2}$

Eine etwas genauere Interpretation des BMI kann mit Tabellen 3 und 4 erfolgen:

Je nach Alter des Probanden kann ein wünschenswerter BMI ermittelt werden:

Erstelle ein Programm, welches nach Eingabe von Alter, Größe, Gewicht und Geschlecht den BMI berechnet, mit Hilfe der Tabelle 3 interpretiert und den wünschenswerten BMI mit Hilfe der Tabelle 4 ermittelt und ausgibt.

Beispiel:

```
Alter              : 33
Groesse            : 1.65
```

	BMI	
	männlich	weiblich
Untergewicht	< 20	< 19
Normalgewicht	20–25	19–24
Übergewicht	25–30	24–30
Fettsucht (Adipositas)	30–40	30–40
massive Fettsucht	> 40	> 40

Tabelle 3: etwas genauerere Interpretation des BMI

Altersgruppe	wünschenswerter BMI
19–24	19–24
25–34	20–25
35–44	21–26
45–54	22–27
55–64	23–28
> 64	24–29

Tabelle 4: Wünschenswerter BMI

```
Gewicht             : 58.7
Geschlecht          : m

Ausgabe:

BMI                 : 21.539
                    : Normalgewicht
wünschenswerter BMI: 20...25
```

2.22 Datumskontrolle ohne Schaltjahr

Erstelle ein Programm, welches nach Eingabe von Tag (1-31), Monat (1-12) kontrolliert, ob das Datum möglich ist. So ist zum Beispiel der 31.04. kein gültiges Datum. *Schaltjahre werden ignoriert!*

2.23 Datumskontrolle mit Schaltjahr

Erstelle ein Programm, welches nach Eingabe von Tag (1-31), Monat (1-12) und Jahr (1583-? / 4-stellig) kontrolliert, ob das Datum möglich ist. So ist zum Beispiel der 31.04.2004 kein gültiges Datum.

Hinweis: Jedes durch 4 teilbare Jahr ist ein Schaltjahr.

Ausnahme: alle durch 100 teilbaren Jahre sind keine Schaltjahre

Ausnahme der Ausnahme: alle durch 400 teilbaren Jahre sind entgegen der Ausnahme doch wieder Schaltjahre!

2.24 Folgedatum

Erstelle ein Programm, welches nach Eingabe von Tag (1-31), Monat (1-12) und Jahr (1583-? / 4-stellig) das richtige Folgedatum berechnet.

2.25 Fahrzeugtyp

Erstelle ein Programm, welches anhand der Eckdaten eines Fahrzeugs eine Kategorisierung durchführt.

1. Stadtwagen: Das Fahrzeug muss weniger als 15.000,-€ kosten, muss mindestens 140 km/h schnell sein und muss weniger als 8 Liter Benzin verbrauchen.
2. Limousine: Das Fahrzeug darf höchstens 25.000,-€ kosten, muss schneller als 190 km/h sein, darf nicht mehr als 10 Liter Benzin verbrauchen und muss mehr als 2 Türen haben.
3. Sportwagen: Das Fahrzeug muss mehr als 50.000,-€ kosten, schneller als 220 km/h sein, sowie 2 Türen besitzen. Der Benzinverbrauch spielt keine Rolle.

Verwende wo möglich boolsche Variablen! Es kann auch sein, dass einige Fahrzeuge nicht eindeutig zugeordnet werden können.

```
Fahrzeugtyp
***********

Preis          :12000
Geschwindigkeit:200
Verbrauch      :14
Kat j/n        :j
Anzahl Türen   :2

Dieses Auto fällt in die Kategorie ......

Programm Wiederholen/Ende (Wiederholen=1,Ende=0) 0
```

2.26 Frage zum Problem Fahrzeugtyp

Gibt es Fahrzeuge, welche in mehr als eine Kategorie fallen? Wenn ja, versuche alle möglichen Kombinationen zu nennen.

2.27 Knobelspiel

Zwei Spieler geben unabhängig voneinander gleichzeitig je eine nicht negative ganze Zahl an (etwa durch Ausstrecken von Fingern auf Kommando oder durch verdecktes Aufschreiben). Nennen beide Spieler die gleiche Zahl, so endet das Spiel unentschieden; andernfalls gewinnt, falls die Summe der genannten Zahlen gerade ist, der Spieler, der die kleinere Zahl genannt hat, und sonst (falls also die Summe ungerade ist) derjenige, der die größere Zahl genannt hat.

Erstelle ein Programm, welches zuerst die zwei gewählten Zahlen einliest und dann am Bildschirm ausgibt, welcher der beiden Spieler gewonnen hat.

2.28 Berechnung der Einkommensteuer einfach

Schreibe ein Programm, welches zu einem eingegebenen Betrag die Einkommensteuer berechnet. Bei weniger oder gleich 5.000,-€ Einkommen wird keine Steuer fällig, ab 5000€ Einkommen wird 20 % auf den Betrag der 5.000,-€ übersteigt, eingehoben.

Beispiel:

Einkommen: 7.500,-

Steuer: 0 % von 5.000,- und 20 % von 2.500,- $= 5000 * \frac{0}{100} + 2500 * \frac{20}{100} = 500$

2.29 Berechnung Einkommensteuer ein bisschen weniger einfach

Erstelle ein Programm, welches zu einem eingegebenen Betrag die Einkommensteuer berechnet. Bei weniger oder gleich 5.000€ Einkommen wird keine Steuer fällig, zwischen 5.000 und 10.000,-€ 20 % (auf den Betrag der 5.000,-€ uebersteigt bis maximal 10.000,-€), ab 10.000,-€ 30 % auf den Betrag der 10.000,-€ übersteigt.

Beispiel:

Einkommen: 15000

Steuer: 30% von 5000 + 20% von 5000 + 0% von 5000 = 1500+1000+0=2500

2.30 Berechnung der Einkommensteuer IRPEF / Einkünfte aus 2002

Erstelle ein Programm, welches die Einkommensteuer (IRPEF) einer natürlichen Person berechnet. Verwende folgende Steuertabelle:

Reddito		Imposta dovuta
oltre	fino a	
0	10.329,14 €	18 % sull'intero importo
10.329,14 €	15.493,71 €	1.859,25 € + 24 % parte eccedente 10.329,14 €
15.493,71 €	30.987,41 €	3.098,75 € + 32 % parte eccedente 15.493,71 €
30.987,41 €	69.721,68 €	8.056,73 € + 39 % parte eccedente 30.987,41 €
69.721,68 €		23.163,10 € + 45 % parte eccedente 69.721,68 €

Tabelle 5: Steuersätze IRPEF 2002

Beispiel:

Bei 20.000,-€ Einkommen

Irpef ist 3.098,75 € + (20.000,-€ - 15.493,71 €) * 32 % = 4.540,76 €

2.31 Osterdatum nach Gauss

Erstelle ein Programm, welches das Osterdatum berechnet. Es gilt folgende Formel:

j=100p+n=Jahreszahl (0¡=n¡100)
q=ganzzahliger Teil von p/3
r=ganzzahliger Teil von p/4
x=Dreißigerrest von 15+p-q-r
y=Siebnerrest von p-r+4
a=Neunzehnerrest von j
b=Viererrest von j
c=Siebnerrest von j
d=Dreißigerrest von 19a+x
e=Siebnerrest von 2b+4c+6d+y

Der Ostersonntag ist am (22+d+e). März oder am (d+e-9). April, je nachdem ob (22+d+e) ¡=31 ist oder nicht.

Ausnahmen:

- Ist d=29 und e=6, so ist Ostern am 19. April.
- Ist d=28 und e=6, so ist Ostern am 18. April.

2.32 Berechnung der Stromkosten

Wandle das Struktogramm der Abbildung 1 in ein Programm um. Achte dabei auf genaue Umsetzung.

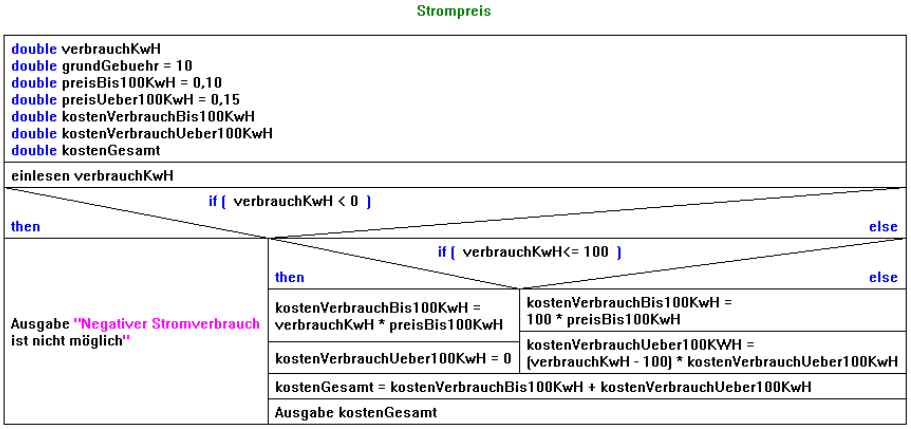

Abbildung 1: Berechnung Stromkosten

2.33 Vergleich Telefonkosten

Wandle die Struktogramme der Abbildungen 2, 3, 4 in Programme um. Achte dabei auf genaue Umsetzung.

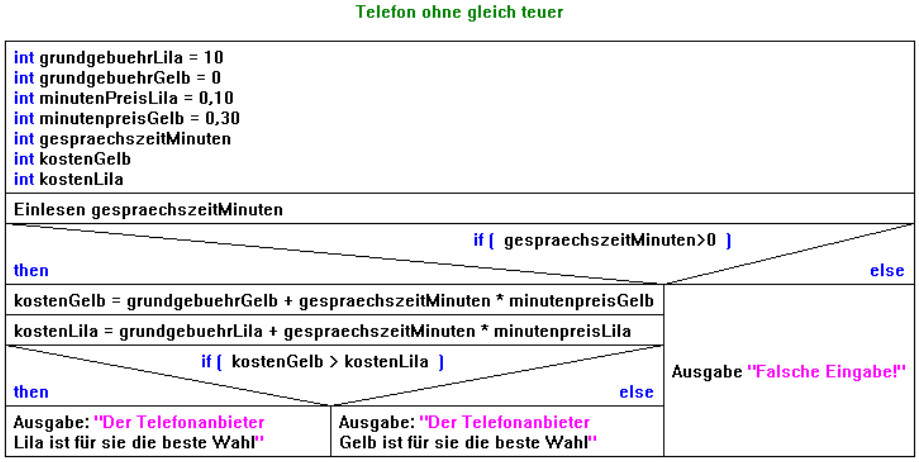

Abbildung 2: Vergleich Telefonkosten ohne gleich teuer

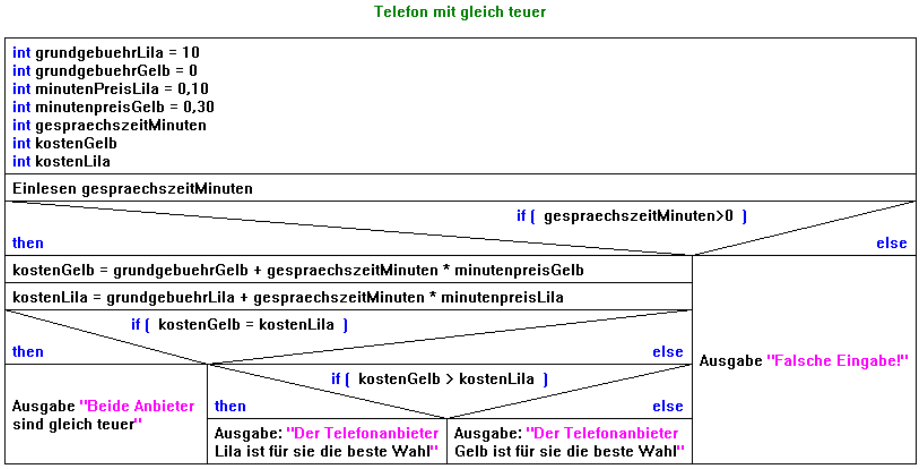

Abbildung 3: Vergleich Telefonkosten mit gleich teuer

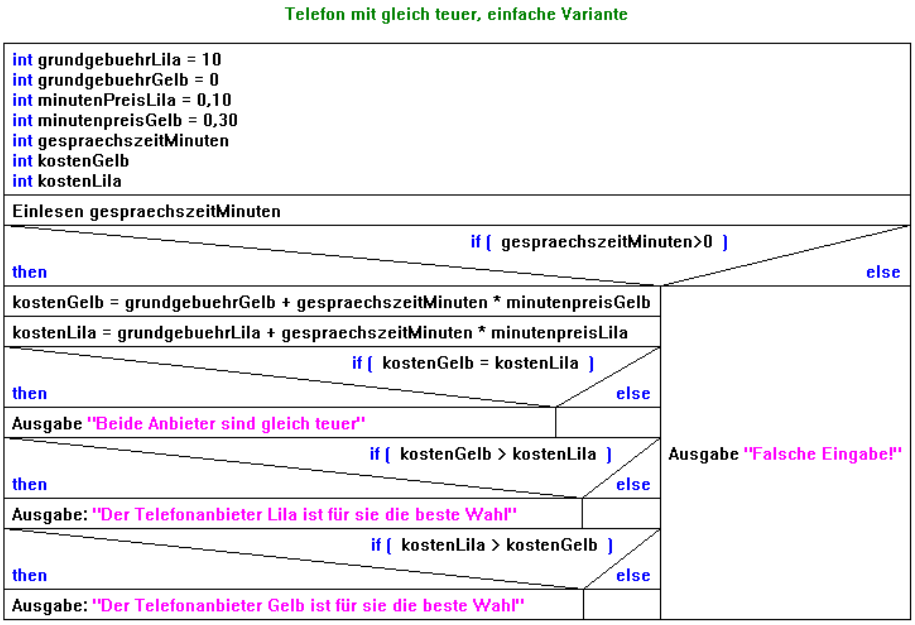

Abbildung 4: Vergleich Telefonkosten mit gleich teuer, einfache Version

2.34 Kleinste Zahl von 3 Zahlen ermitteln

Schreibe ein Programm, welches für drei eingegebene Zahlen die kleinste Zahl ermittelt und am Bildschirm ausgibt.

2.35 3 Zahlen der Größe nach ausgeben

Schreibe ein Programm, welches drei eingegebene Zahlen der Größe nach sortiert am Bildschirm ausgibt.

2.36 Rest Getränkeautomat

Entwirf ein Programm für einen Nahrungsmittel-Automaten, welches errechnet, welche Restmünzen in welcher Anzahl zurückgegeben werden müssen. Der Preis der Speisen ist immer ein Vielfaches von 0,10 €. Der Automat akzeptiert Geldscheine zu 5,- € und 10,- €, Münzen zu 0,10 €, 0,20 €, 0,50 €, 1,- € und 2,- €. Der Restbetrag wird nur in Münzen ausbezahlt (maximal 3,- €). Nach Angabe der Höhe des eingeworfenen Geldes und des Preises der ausgewählten Waren soll errechnet werden, welche Münzen in das Restfach fallen müssen. Das Programm soll auch eine Kontrolle durchführen, ob die eingegebenen Beträge überhaupt zulässig sind.

Zusatz: Es kommt öfters vor, dass der Vorrat an 2,- € Münzen erschöft ist. In diesem Falle sollen zwei 1,- € Münzen an Stelle einer 2,- € Münze ausbezahlt werden. Simuliere die Situation in der keine 2,- € Münzen verfügbar sind.

2.37 Lösung Quadratische Gleichung

Quadratische Gleichungen können mit der Formel $x_{1,2} = -b \pm \frac{\sqrt{b^2-4ac}}{2a}$ gelöst werden. Schreibe ein Java-Programm, das die Werte a, b und c vom Typ **double** vom Benutzer einliest und die Lösungen x1, x2 der Gleichung berechnet. Bei nicht berechenbaren Lösungen soll der Benutzer eine aussagekräftige Meldung erhalten. Hinweis: Math.sqrt(zahl) berechnet die Wurzel aus zahl.

3 Schleifen

3.1 Ausgabe wiederholen

Erstelle ein Programm, welches 19mal untereinander „Bald schlafe ich ein...“ ausgibt.

3.2 Sternchen 1

Ausgabe von n Sternen nebeneinander am Bildschirm ausgibt. n wird vom Benutzer eingegeben.

Beispiel:

Wie viele Sterne: 10

```
**********
```

3.3 Sternchen 2

Aus Sternen soll ein Quadrat am Bildschirm erstellt werden. Die Seitenlänge wird vom Benutzer eingegeben.

```
***
***
***
```

3.4 Sternchen 3

Aus Sternen soll ein Rahmen am Bildschirm erstellt werden. Beide Seitenlängen werden vom Benutzer eingegeben.

```
**********
*        *
*        *
**********
```

3.5 Sternchen 4

Aus Sternen soll eine Dreieck am Bildschirm erstellt werden. Der Benutzer gibt die Höhe (oder Breite) ein.

```
*
**
***
****
```

3.6 Sternchen 5

Aus Sternen soll eine Pyramide am Bildschirm erstellt werden. Der Benutzer gibt die Höhe ein.

```
   *
  ***
 *****
*******
```

3.7 Sternchen 6

Wie Sternechen 5 jedoch gibt der Benutzer die Breite der untersten Reihe an. Achte darauf, dass nur ungerade Eingaben sinnvoll sind.

Zusatz: (do-while-Schleife) Die Eingabe soll so lange wiederholt werden, bis eine gültige Eingabe getätigt worden ist.

3.8 Zahlen zwischen 1 und 100

Alle Zahlen zwischen 1 und 100 sollen nebeneinander am Bildschirm ausgegeben werden.

3.9 Zahlen zwischen 1 und Obergrenze

Alle Zahlen zwischen 1 und einer vom Benutzer einzugebenden Obergrenze sollen (nebeneinander, durch ein Komma getrennt) am Bildschirm ausgegeben werden.

3.10 Zahlen zwischen 1 und Obergrenze ohne letztes ,

Die Bildschirmausgabe des Programms aus Aufgabe 3.9 wird wahrscheinlich folgendes Aussehen haben:

```
1,2,3,4,5,
```

Verändere das Programm so, dass das letzte Komma nicht am Bildschirm ausgegeben wird.

3.11 Zahlen zwischen 10 und 30

Alle Zahlen zwischen 10 und 30 sollen nebeneinander am Bildschirm ausgegeben werden.

3.12 Durch 2 teilbar?

Alle durch 2 teilbaren Zahlen zwischen 10 und 999 sollen am Bildschirm ausgegeben werden.

Verwende hierzu zwei Lösungsansätze

1. Schleife mit Schrittweite 2
2. Schleife mit Schrittweite 1; Kontrolle ob Schleifenzähler gerade oder ungerade ist (modulo Operator %) und ggf. Ausgabe

3.13 Was berechnet dieses Programm?

Was berechnet das folgende Programm?

```
1  import corejava.*;
2
3  public class fragezeichen2
4  {
5    static public void main (String[] arguments)
6    {
7      long n,i;
8      double summe;
9      double eingabe;
10     double ergebnis;
11
12
13     summe = 0.0;
14
15     for(i=0;i<=5;i++)
16     {
17       eingabe = Console.readInt("Geben sie nun ..... ein: ");
18       summe=summe+eingabe;
19     }
20     ergebnis = summe / 5;
21     System.out.print("Ihr Ergebnis betraegt: "+ ergebnis);
22   }
23 }
```

Tipp: Es handelt sich um eine von Schülern sehr häufig verwendete Kenngröße!

3.14 Programm wiederholen

Verändere 3 bereits erstellte Programme deiner Wahl so, dass das Programm am Ende wiederholt werden kann. Dabei erhält der Benutzer die Frage

`Willst du das Programm wiederholen (1=ja, 2=nein)?`

Antwortet der Benutzer mit 1 soll das Programm wiederholt werden, antwortet er mit 2 soll das Programm beendet werden.

3.15 Benutzereingaben überprüfen

Verändere die Benutzereingaben bei 3 bereits erstellten Programmen deiner Wahl so, dass die Benutzereingaben so lange wiederholt werden, bis die Eingabe korrekt ist.

Beispiel Einkommensteuerberechnung:

```
Geben sie das Einkommen ein: -2000
Fehler: negative Einkommen sind nicht zulässig!
Geben sie das Einkommen ein: -100
Fehler: negative Einkommen sind nicht zulässig!
Geben sie das Einkommen ein: 20000
Die Einkommensteuer IRPEF beträgt: xxxxx
```

3.16 Raucherproblem mit Wiederholung und Eingabenkontrolle

Verändere das Programm „Blauer Dunst“ 1.19 wie im Zusatz gefordert.

3.17 while-Schleife aus for-Schleife

Ersetze die for-Schleife aus dem Programm der Aufgabe 3.13 durch eine while-Schleife.

3.18 while-do zu do-while 1

Ersetze in folgendem Programm die while-do Schleife durch eine do-while Schleife

```
1  import corejava.*;
2  public class whiledo2dowhile1
3  {
4    public static void main (String[] args)
5    {
6      String name="";
7      while (! name.equals("Rumpelstilzchen"))
8      {
9        name = Console.readString("Errate␣meinen␣Namen");
10     }
11     System.out.println("Das␣hat␣dir␣der␣Teufel␣gesagt...");
12   }
13 }
```

3.19 while-do zu do-while 2

Ersetze in folgendem Programm die while-do Schleife durch eine do-while Schleife

```
1  import corejava.*;
2  public class whiledo2dowhile2
3  {
4    public static void main (String[] args)
5    {
```

```
6       String wiederholen="j";
7       while (wiederholen.equals("j"))
8       {
9         System.out.println("gone to do something...");
10        wiederholen = Console.readString("Programm wiederholen (j,J) ?");
11        wiederholen = wiederholen.toLowerCase();
12      }
13    }
14  }
```

3.20 do-while zu while-do

Ersetze in folgendem Programm die do-while Schleife durch eine while-do Schleife

```
1  import corejava.*;
2  public class dowhile2whiledo
3  {
4    public static void main (String[] args)
5    {
6      int eingabe;
7      do
8      {
9        eingabe = Console.readInt("Gib dein Alter ein: ");
10       if (eingabe < 0 || eingabe >= 99)
11       {
12         System.out.println("Du bist zu alt oder zu jung für dieses Spiel...");
13       }
14       else
15       {
16         System.out.println("OK...");
17       }
18     } while (eingabe<=0 || eingabe >= 99);
19   }
20 }
```

3.21 Überprüfung auf Quadratzahl

Überprüfe, ob eine Zahl n eine Quadratzahl ist, indem du, beginnend bei 1, alle ungeraden Zahlen aufsummierst. Wenn sie in der Summe die Zahl n erreichen, so ist n eine Quadratzahl, sonst nicht.

Beispiel:

```
n: 9
Die Zahl ist eine Quadratzahl

n: 11
Die Zahl ist keine Quadratzahl
```

3.22 Potenz

Erstelle ein Programm, welches b^e berechnet und am Bildschirm ausgibt. Die Basis b ist ein **double**-Wert, der Exponent e eine Ganzzahl.

3.23 Statistik: Münzwurf - mehrere Programme

- Feststellen von relativen und absoluten Häufigkeiten.
- Angabe der längsten ununterbrochenen Folge der selben Münzseite: alle vorkommenden Folgen von gleichen Münzseiten kategorisieren (1* hintereinander, 2* hintereinander, 3* usw. bis zum Maximum) mit den jeweiligen absoluten und relativen Häufigkeiten)
- Überprüfe, ob die aus der Mathematik bekannte Wahrscheinlichkeitsverteilung mit unseren Simulationsergebnissen übereinstimmt.

3.24 Zahl halbieren

Wie oft muss eine Zahl durch 2 geteilt werden, damit als Ergebnis eine Zahl kleiner 10 herauskommt? Verwende eine while-Schleife und eine Variable für die Anzahl der Divisionen. Kommentiere die Lösung und Erläutere deine Entscheidung für die do-while-Schleife oder die while-Schleife.

```
Halbieren
*********

Zahl: 99999
Anzahl Divisionen durch 2: 14

Programm Wiederholen/Ende (Wiederholen=1,Ende=0) 0
```

3.25 Berechnung der Kreiszahl PI I

Der Mathematiker Leibniz hat herausgefunden, dass sich die Kreiszahl π folgendermaßen annähernd berechnen läßt:
$\frac{\pi}{4} = 1 - \frac{1}{3} + \frac{1}{5} - \frac{1}{7} + \frac{1}{9} - \frac{1}{11} + \ldots$
Erstelle einen Algorithmus, der diese Berechnungsmethode verwendet um π näherungsweise zu berechnen.
Beispiel:

```
Berechnung der Zahl Pi
**********************

Wie viele Rechendurchläufe sollen durchgeführt werden? 1000

PI = ????

Programm Wiederholen/Ende (Wiederholen=1,Ende=0) 0
```

An Stelle der vier Fragezeichen soll das Ergebnis ausgeben werden.

3.26 Berechnung der Kreiszahl PI II

Etwa zur gleichen Zeit fand der Mathematiker John Wallis eine andere Methode zur Berechnung von π.
$\frac{pi}{2} = \frac{2}{1} * \frac{2}{3} * \frac{4}{3} * \frac{4}{5} * \frac{6}{5} * \frac{6}{7} * \ldots$
Dein Programm soll π nach beiden Verfahren jeweils bis zur angegebenen Anzahl an Berechnungsschritten berechnen. Die beiden Ergebnisse werden mit einem Verweis auf die verwendete Methode ausgegeben, wobei jeweils zusätzlich der Unterschied zur von Java bereitgestellten Kreiszahl π (dem Wert Math.PI) ausgegeben wird.

3.27 Harmonische Reihe

Bestimme die Summe der n ersten Glieder der harmonischen Reihe $s = \frac{1}{1} + \frac{1}{2} + \frac{1}{3} + .. \frac{1}{n}$.
Varianten:

1. Bestimme die Summe der n ersten Glieder der harmonischen Reihe $s = \frac{1}{1} - \frac{1}{3} + \frac{1}{5} - \frac{1}{7} + \frac{1}{9} - .. + -1^{n-1} * \frac{1}{n^{2n-1}}$. Bemerkung: Diese Reihe konvergiert gegen $\frac{\pi}{4}$.
2. Der Parameter, der eingegeben wird, soll nun angeben, wie stark das Ergebnis von $\frac{\pi}{4}$ abweichen darf. Bestimme sie bis zu welchem Folgenglied die Reihe berechnet werden muss, damit dieses Genauigkeit mindestens erreicht wird.
3. Der eingegebene Parameter soll nun die Anzahl der Stellen angeben, die bei der Reihensumme mit dem Grenzwert übereinstimmen müssen. Tipp: Zwei Zahlen a und b sind in den ersten (n-1) bis n Stellen gleich, wenn $|a - b| < |a| * 10^{-n}|$ ist., d.h. der relative Fehler ist 10^{-n}

3.28 Quadratwurzel näherungsweise berechnen

Erstelle ein Programm, welches die Quadratwurzel einer natürlichen Zahl z annähert. Benutze dazu die Folge von Heron:

$$X_{n+1} = \frac{1}{2} * (X_n + \frac{X_0}{X_n})$$

Das Programm soll abbrechen, wenn der Unterschied zum vorherigen Näherungswert kleiner als 0.0000000001 wird. Der Wert X_0 ist die Zahl selbst.

3.29 Radioaktiver Zerfall

Radioaktive Stoffe zerfallen mit einer bestimmten Halbwertszeit, das heisst nach dieser Zeit ist die ausgehende radioaktive Strahlung nur noch halb so stark. Berechne die Restaktivität nach einer variablen Anzahl von Jahren.

Beispiel:

```
Halbwertzeit
**********

Halbwertzeit (Jahre) : 12000
Restaktivität in wie viel Jahren: 100
Restaktivität nach 100 Jahre: 99,58 %

Programm Wiederholen/Ende (Wiederholen=1,Ende=0) 0
```

3.30 AfA - linear

Erstelle eine Abschreibungstabelle eines Anlageguts. Jedes Jahr wird ein gleichbleibender Betrag abgeschrieben. Der Benutzer soll die Dauer der Abschreibung eingeben, der Abschreibungssatz soll berechnet werden. Beispiel:

Wert: 10000

Abschreibungsdauer (Jahre): 10

Abschreibungssatz: 10 %

Jahr	Abschreibung	Restewert
1	1.000	9.000
2	1.000	8.000
3	1.000	7.000
...		
10	1.000	0

Tabelle 6: Beispiel lineare Abschreibungstabelle

3.31 AfA - degressiv

Erstelle eine Abschreibungstabelle eines Analgeguts. Jedes Jahr wird dabei ein Prozentsatz des Restwerts abgeschrieben. Der Benutzer gibt den Wert des Guts und den %-Satz der Abschreibung ein. Die Tabelle soll für 15 Jahre berechnet werden.

Beispiel: Wert: 10000

Abschreibungssatz: 10 %

Jahr	Abschreibung	Restewert
1	1.000	9.000
2	900	8.100
3	810	7.000
...		
14	254,19	2287,68
15	228,77	2058,91

Tabelle 7: Beispiel degressive Abschreibungstabelle

3.32 Dezimalzahl zu dualzahl

Erstelle ein Programm, welche eine Dezimalzahl in eine Dualzahl umrechnet. Die Ausgabe der Dualzahl darf dabei auch in umgekehrter Reihenfolge erfolgen.

3.33 Teiler

Erstelle ein Programm, welches alle Teiler einer vom Benutzer eingegebenen Zahl zwischen 1 und der Zahl selbst am Bildschirm ausgibt.

```
Von welcher Zahl willst du die Teiler? 15

Die Teiler sind:
1,3,5,15
```

3.34 Primzahlkontrolle

Eine Primzahl ist eine Zahl, die nur durch eins und durch sich selbst ohne Rest teilbar ist. Erstelle ein Programm, das überprüft, ob es sich bei einer eingegebenen Zahl um eine Primzahl handelt oder nicht.

Beispiel:

```
Gib die zu überprüfende Zahl ein: 9
9 ist keine Primzahl
```

Zusatz: Optimierung der Primzahlkontrolle:

1. Um festzustellen, ob die Zahl n eine Primzahl ist, genügt die Kontrolle der Division ohne Rest durch alle Zahlen zwischen 1 und $\sqrt{n} + 1$.
2. Verwende zusätzlich eine Abbruchbedingung um die Schleife vorzeitig abzubrechen, wenn wir festgestellt haben, dass es sich nicht um eine Primzahl handelt.

3.35 Zinseszinsrechnung

Erstelle ein Programm, welches nach Angabe von Kapital, Zinssatz, Jahre das Endkapital mit Zinseszins berechnet. Erstelle hierzu zwei Lösungen:

1. mit Zinseszinsformel $z = (1 + \frac{p}{100})^n$. Verwende zur Berechnung der Potenz die Methode Math.pow(basis, exponent).
2. mit wiederholter Zinsberechnung in einer Schleife

3.36 Werde ich betrogen?

Dir wird die Teilnahme an folgendem Würfelspiel angeboten: Gespielt wird mit zwei Würfeln mit den Augen 1 bis 6, deren Augensumme addiert wird. Dein Einsatz beträgt bei jedem Spiel 0,50 €. Dein Gewinn berechnet sich nach der untenstehenden

Auszahlungstabelle. Würdest du an diesem Spiel teilnehmen? Mit anderen Worten wirst du auf lange Sicht eher Geld verlieren oder gewinnen?

Erstelle ein Java-Programm, das das Würfelspiel simuliert und dir deinen Gesamtgewinn (oder Verlust) nach 100.000 Spielen anzeigt.

Augensumme	Auszahlung	Gewinn
12	4-facher Einsatz	+1,50 €
11	3-facher Einsatz	+1,00 €
10	2-facher Einsatz	+0,50 €
7,8,9	Einsatz zurück	+0,00 €
2,3,4,5,6	keine	-0,50 €

Tabelle 8: Gewinnplan des Spiels

3.37 Gleichverteilung von Math.random()

Teste die Gleichverteilung der von Math.random() erzeugten Zufallszahlen. Simuliere dazu das Werfen eines Würfels mit den Augenzahlen 1 bis 6. Würfle mindestens 100.000 mal. Alle 10.000 Würfe soll die relative Häufigkeit der Augenzahlen ausgegeben werden. Beispiel für eine Ausgabe:

```
Ich würfle 100000 mal. Die relative Häufigkeit beträgt
nach n Würfen      1      2      3      4      5      6
   10000          0,17   0,18   0,13   0,14   0,17   0,21
   20000          0,12   0,155  0,165  0,175  0,15   0,235
   ...
   100000         0,165  0,175  0,177  0,169  0,158  0,156
```

Zusatz: würfle mit zwei Würfeln. Beachte dabei, dass ein Würfeln mit zwei Würfeln nicht das Selbe ist wie mit einem zwölferwürfel zu würfeln. Entsprechen die Ergebnisse deinen Vermutungen?

3.38 Wildbestand

In unseren Wäldern gibt es keine Wölfe, Luchse und andere große Raubtiere mehr. Das einzige „Raubtier“, das für die Regulation des Wildbestandes sorgt, ist der Mensch. Ohne ihn würden sich Rehe stärker vermehren, als es für den Wald gut wäre.

Der Bestand hängt von der Anfangspopulation, der Vermehrungsrate und der jährlichen Abschußrate ab.

Entwirf und formuliere einen Algorithmus, der zur Berechnung dient, wie sich der Bestand an Rehen jährlich entwickelt.

Erweitere den Algorithmus so, dass er den Bestand für die nächsten Jahre berechnet und am Bildschirm ausgibt. Verwende hierzu eine while-Schleife. Die Berechnung soll gestoppt werden, wenn die Obergrenze von 300 Rehen erreicht ist oder wenn alle Rehe ausgerottet sind.

Verwende Konstanten für den Anfangsbestand 200 und Vermehrungsrate 10%. Die Abschussrate wird vom Benutzer eingegeben.

3.39 Ein- Mal- Eins- Tabelle

Es soll eine Tabelle erstellt werden, die in der ersten Zeile (Kopfzeile) und der ersten Spalte (Kopfspalte) jeweils die Zahlen 1 bis 20 enthält und in der so entstandenen Matrix das Produkt aus der Zahl in der Kopfzeile und Kopfspalte.

Entwickle einen Algorithmus mit Hilfe von Schleifen um diese Tabelle zu erstellen. Es empfiehlt sich zwei Schleifen ineinander zu verschachteln. Die äußere ist für die Zeilen zuständig, die innere für die Werte in der einzelnen Spalten.

3.40 Quersumme

Schreibe ein Programm, welches die Quersumme einer ganzen Zahl berechnet. Die Quersumme einer Zahl ist die Summe der Dezimalziffern der Zahl. Beispielweise hat die Zahl -123456 die Quersumme 21. Hinweis: verwende die ganzzahlige Division und ganzzahlige Restbildung(%).

3.41 Fakultät

Schreibe ein Programm zur Berechnung der Fakultät einer Zahl n mit einer while-Schleife. Die Fakultät einer Zahl n berechnet sich wie folgt:

$$n! = 1 * 2 * 3 * 4 * 5 * 6 * 7 * \ldots * n$$

3.42 Pythagoräische Zahlentripel

Pythagoräische Zahlentripel sind alle natürlichen Zahlentripel, bei denen die Summe der Quadrate zweier Zahlen a und b das Quadrat der dritten Zahl c ergibt: $a^2 + b^2 = c^2; (a < b < c)$ Erstelle ein Java-Programm, das alle pythagoräischen Zahlentripel bis zu einer oberen Grenze $c = 100$ ausgibt. Verwende geeignete Schleifen und vermeide Mehrfachausgaben. Ihre Tabelle sollte mit folgenden Zahlentripeln beginnen (nach aufsteigendem c geordnet):

3	4	5
6	8	10
5	12	13
9	12	15
8	15	17

Tabelle 9: Pythagoräische Zahlentripel bis c=17

4 Felder

4.1 Zufallszahlen

Erstelle ein Feld mit 2.300 Elementen vom Typ **int**. Im Feld werden Zufallszahlen im Bereich zwischen -5 und 67 (inklusive) gespeichert. Gib anschliessend das Feld am Bildschirm aus.

4.2 Wuerfelspiel

Erstelle ein Feld mit 5.000 **byte**-Elementen. Fülle alle 5.000 Elemente des Feldes zufällig mit Werten von 1 bis 6. Berechne, wie oft jeder der möglichen Werte (1,2,3,4,5,6) vorkommt, und stelle anschliessend die Ergebnisse als absolute Zahl und in Prozenten dar (absolute und relative Häufigkeiten). Das Erzeugen der Zufallszahlen und die Berechnung der Häufigkeiten müssen in getrennten Schleifen erfolgen!

Ausgabe am Bildschirm:

```
Nach 5000 mal wuerfeln erhalten wir folgende Ergebnisse:
1er: 818 entspricht 16.36 %
2er: 841 entspricht 16.82 %
3er: 818 entspricht 16.36 %
4er: 866 entspricht 17.32 %
5er: 814 entspricht 16.28 %
6er: 843 entspricht 16.86 %
```

4.3 Zufallszahlen erzeugen, Minimum-Maximumsuche

Erzeuge ein Feld variabler Größe. Der Benutzer gibt ein, wie viele Zufallszahlen erzeugt werden sollen. Die Zufallszahlen sollen im Bereich zwischen 18 und 67 (jeweils einschließlich) erzeugt und im Feld gespeichert werden.

1. Der Benutzer soll den Bereich der Zufallszahlen selbst wählen können.
2. Nun soll das Maximum, das Minimum und der Durchschnitt der erzeugten Zahlen berechnet werden.
3. Gib *die Stellen* aus, an welchen sich das Maximum bzw. das Minimum befindet.
4. Der Inhalt des Feldes soll umgedreht (also der Inhalt der ersten Stelle an die letzte, der Inhalt der zweiten Stelle an die vorletzte Stelle usw.) werden.

4.4 Figurella

Das Schlankheitsstudio "Figurella"verspricht seinen Kunden, dass sie, wenn diese den strikten Diätplan einhalten, in den ersten 10 Monaten jeden Monat 3% ihres Körpergewichts verlieren, danach jeden Monat 2%: du sollst nun ein Programm

erstellen, das als Eingaben das ursprüngliche Körpergewichts sowie die Dauer der Abmagerungskur in Monaten erhält und dann für jeden Monat das Körpergewicht des Kunden in einem Feld (dessen Größe du dir selbst überlegen musst) speichert. Am Schluss soll der Inhalt dieses Feld folgendermaßen ausgegeben werden.

```
Monat Gewicht
1     xxx
2     xxx
3     xxx
```

4.5 Verlorene Noten

Ein Mathematiklehrer möchte die Ergebnisse der letzten Schularbeit in sein Register übertragen. Der kleine Sohn des Lehrers hat sich jedoch einen Scherz erlaubt: er hat die Noten durcheinandergebracht und dem Vater nur den Lösungsweg aufgezeigt. Da der zerstreute Lehrer gerade Java lernt, nutzt er die Gelegenheit, ein Programm zur Berichtigung zu erstellen und gleichzeitig einige statistische Auswertungen durchzuführen.

Er benutzt ein Feld mit dem Namen Noten und initialisiert dieses mit den verfälschten Ergebnissen. Der Lehrer merkt jedoch, daß seine Kenntnisse nicht ausreichen und gibt auf.

Vervollständige das Programm wie folgt:

1. Zu jedem Element soll die Zahl 5 hinzugezählt werden.
2. Jedes 2. Element soll durch die Zahl 2 dividiert werden.
3. Erstelle ein zusätzliches Feld (Name: notenKopie) und kopiere alle Werte in dieses neue Feld
4. Erstelle ein 3. Feld (Name: notenRueckwaerts) und kopiere alle Werte in umgekehrter Reihenfolge
5. Berechne den Mittelwert (Summe aller Zahlen geteilt durch Anzahl!) der Zahlen in diesem Feld. Frage: ist der Mittelwert aller Felder gleich oder nicht?
6. Berechne das Maximum und das Minimum.

```
1  // Aufgabe
2  //
3  //
4
5  import corejava.*;
6  class schularbeit
7  {
8    static void main(String[] args)
9    {
10     int i;
11     int j;
12     int noten[]={5,10,3,7,2,4,2,10,1,7};
13     int notenKopie[]=new int[10];
14     int notenRuaeckwerts[]=new int[10];
15
```

```
16
17      }
18  }
```

4.6 Zufallszahlen II

Erstelle ein Feld (Array) mit 100 Elementen vom Typ **double**. Im Feld werden Zufallszahlen im Bereich zwischen $0[$ und $1]$ gespeichert. Finde die kleinste Zahl und gib sie aus.

4.7 Zufallszahlen III

Wie könnte die Aufgabe 4.6 ohne Feld gelöst werden?

4.8 Suche nach einer Zahl im Feld

Fülle ein Feld mit 1000 Zufallszahlen zwischen 0 und 100. Der Benutzer gibt eine Zahl ein und der Rechner überprüft, ob die eingegebene Zahl unter den erzeugten Zufallszahlen ist.

Wie könnte dieses Programm ohne Feld geschrieben werden?

Alternativen

- Berechne ebenfalls, wie oft die Zahl vorgekommen ist.
- Die Positionen, an denen sich die Zahl befindet, sollen auch ausgeben werden.

4.9 Primzahlen mit dem Sieb des Erathostenes

Das Berechnen der Primzahlen hat Mathematiker und Philosophen schon immer fasziniert. So auch den Griechen Eratosthenes (275 v.Chr. - 194 v.Chr an den Folgen eines Hungerstreiks), der dazu ein nach ihm benanntes Verfahren entwickelte. Sein eigentlich viel größerer Verdienst war die Bestimmung des Erdradiuses zu 250000 Stadien. Hätte Columbus mehr griechische Bildung gehabt, so hießen die Indianer heute nicht Indianer ;-)

Um die Primzahlen zu bestimmen schrieb Eratosthenes alle Zahlen von 2 bis zu der gewünschten Größe auf und strich dann bei 2 beginnend alle Vielfachen von 2 aus der Liste. Dann ging er zur nächsten Zahl in der Tabelle - 3 - und strich ebenfalls alle Vielfachen von 3. Die nächste noch vorhandene Zahl war nun 5, deren Vielfache wiederum aus der Liste gestrichen werden. So fuhr er fort, bis er am Ende der Liste angelangt war (so weit muss man es nicht machen). Die übrig gebliebenen Zahlen sind dann Primzahlen.

Dein Java-Programm soll nun vom Benutzer abfragen bis zu welcher Zahl die Primzahlen berechnet werden sollen. Dann werden diese mit Hilfe des Siebs von Eratosthenes berechnet und am Bildschirm ausgegeben. Verwende dazu ein eindimensionales Feld.

4.10 Vom Zahlenteiler zu Primzahlen

Bestimme zu einer als Parameter eingegebenen Zahl alle Teiler (alle natürlichen Zahlen von 1 bis n, die die Zahl ohne Rest teilen, benutze hierzu den Modulo-Operator %) und gib diese am Bildschirm aus.

Varianten:

1. Speichere bei Erstellen des Objekts Teiler01 alle Teiler in einem Array mit bei Programmerstellung festgelegter Länge ab und gib die Zahlen anschließend auf dem Bildschirm aus.
2. Da die Länge des nötigen Teiler-Arrays vor Durchlaufen des Algorithmus nicht bekannt ist, ist es evtl. sinnvoll, den Algorithmus zweimal zu durchlaufen, wobei das erste Mal nur gezählt wird, wie viele Teiler die Zahl n hat.
3. Eine Zahl ist eine Primzahl, wenn sie genau zwei Teiler hat. Nutze dies aus, um zu entscheiden, ob eine Zahl eine Primzahl ist oder nicht.
4. Ergänze das obige Programm so, dass es dann, wenn n keine Primzahl ist, die nächsthöhere Primzahl bestimmt.
5. Die Teiler werden nicht in einem Array abgespeichert, sondern in einem Vector.

4.11 Teiler einer Zahl

Erstelle ein Programm, das alle Teiler einer Zahl bestimmt und am Bildschirm ausgibt. Teiler einer Zahl kommen immer in Paaren vor. So wird die Zahl 6543 durch die Zahlen 1, 3, 9, 727, 2191 und 6543 geteilt. die Paare sind dann (1,6543), (3, 2181), (9, 727). Erstelle zwei Programme welche die Teiler einer Zahl berechnen.

1. Alle Zahlen zwischen 1 und der Zahl selbst werden überprüft und in einem Feld gespeichert, anschließend werden die Paare gebildet und am Bildschirm ausgegeben.
2. Verwende die Kenntniss, dass sich jeweils ein Teiler des Teilerpaares aus dem anderen Teiler berechnen lässt, um die Schleife vorzeitig beenden zu können.
3. Führe anschließend einen Vergleich der Rechenzeit beider Programme durch.

4.12 Zweidimensionales Feld füllen

Schreibe ein Programm, das ein zweidimensionales Feld **int** [][] mit n mal m Elementen so initialisiert, dass in der ersten Zeile die Werte (11, 12, 13, ...), in der zweiten die Werte (21, 22, 23) und in der dritten die Werte (31, 32, 33) stehen.

4.13 Überprüfung magisches Quadrat

Magische Zahlenquadrate sind Zahlenfelder, bei denen jede Zahlenreihe, egal ob waagrecht, senkrecht oder diagonal addiert, dieselbe Summe ergibt. Dabei kommt

jede Zahl nur einmal vor. Die Schwierigkeit liegt jedoch in ihrer Anordnung. Das einfachste Quadrat umfasst 3 mal 3, also 9 Felder, kompliziertere 64 oder 81 Felder.

Erstelle ein Programm, welches überprüft, ob es sich bei folgenden Zahlen um ein magisches Zahlenquadrat handelt. Speichere die Zahlen in einem zweidimensionalen Feld ab.

4	9	2
3	5	7
8	1	6

Tabelle 10: Magisches Zahlenquadrat

4.14 Matrix um 90° nach links drehen

Schreibe ein Programm, das eine quadratische (n x n) - Matrix (2D-Feld) um 90° nach links rotiert. Beispiel: 11 wird zu 12.

1	2	3
4	5	6
7	8	9

Tabelle 11: Matrix vor dem Drehen

wird zu

3	6	9
2	5	8
1	4	7

Tabelle 12: Matrix nach dem Drehen

4.15 Pascalsches Dreieck

Das Pascalsche Dreieck ist ein Dreieck aus Zahlen hat folgenden Aufbau:

```
     1          n=0
    1 1         n=1
   1 2 1        n=2
  1 3 3 1       n=3
 1 4 6 4 1      n=4
```

Anschaulich bedeutet das, dass in dem Aufbau des Pascalschen Dreiecks der Wert eines Feldes durch die Summe der schräg rechts und links oberhalb liegenden Werte berehnet wird. Ist der gesuchte Wert am Rand, so ergibt sich per Definition eine 1.

Erstelle ein Programm welches die Zahlen des Pascalschen Dreiecks in einem zweidimensionalen Feld bis zu n=10 speichert.

5 String-Operationen

5.1 Vergleiche

1. Zwei Zeichenketten werden über die Tastatur eingelesen. Am Bildschirm soll ausgegeben werden, ob die beiden Zeichenketten gleich sind oder nicht.
2. Wie im vorhergehenden Beispiel, die Groß- und Kleinschreibung soll jedoch ignoriert werden.

5.2 Buchstaben zählen

Erstelle ein Programm, welches zählt, wie oft ein eingegebener Buchstabe in einem vorgegebenen Text vorkommt. Beispiel:

```
Vorgegebener Text: "eine Uebung"
Eingegebener Buchstabe: "e"
Ausgabe am Bildschirm: Im Text "eine Uebung" kommt "e" 3 mal vor!
```

5.3 Klein- zu Großbuchstaben und umgekehrt

Eine Zeichenkette wird eingelesen. Aus dieser Zeichenkette sollen zwei neue Variablen gebildet werde, welche den eingegebenen Text jeweils in Klein- und Großbuchstaben beinhalten. Diese Variablen sollen am Bildschirm ausgegeben werden.

5.4 Teile von Zeichenketten

Aus einem eingelesenen String werden die Buchstaben 3-6 ausgeschnitten und am Bildschirm ausgegeben. Wenn die Zeichenkette weniger als 6 Buchstaben hat, soll eine Fehlermeldung erscheinen.

5.5 Einzelne Buchstaben

Die Buchstaben einer eingegebenen Zeichenkette sollen untereinander am Bildschirm ausgegeben werden.

5.6 Vokale zählen

Erstelle ein Programm welches die Vokale eines eingegebenen Textes zählt und am Bildschirm ausgibt.

5.7 Buchstaben ersetzen

Erstelle ein Programm, welches in einer vorgegebenen Zeichenkette alle einem eingegebenen Buchstaben entsprechenden Zeichen durch einen Stern (*) ersetzt!

5.8 Text umdrehen

Erstelle ein Programm welche einen eingegebenen Text umdreht. Dabei sollen zwei Vorgangsweisen benutzt werden:

1. Ausgabe der einzelnen Zeichen in umgekehrter Reihenfolge
2. Abspeichern in einer neuen Variable (Tipp: Verkettung der Buchstaben mit +)

5.9 Palindrom

Ein Palindrom ist ein Wort, das von beiden Richtungen gelesen werden kann. Beispiele: otto, anna ...

Erstelle ein Programm, welches überprüft, ob es sich bei einem eingegebenen Text um ein Palindrom handelt.

1. Leerschritte werden berücksichtigt.
2. Leerschritte werden ignoriert. So ist zum Beispiel der Text „go hang a salami im a lasagna hog“ nur von hinten lesbar wenn die Leerschritte ignoriert werden.

5.10 Klammersetzung

Erstelle ein Programm welches einen geklammerten Term, wie er in der Mathematik vorkommt überprüft, ob die Klammersetzung korrekt ist. Es sollen nur Terme mit runden Klammern überprüft werden. Bei falscher Klammersetzung soll auch eine Begründung ausgegeben werden.

Term	richtig / falsch	Begründung
((a+b)*c)	r	
a*(b-c	f	Klammer zu fehlt

Tabelle 13: Beispiele Klammersetzung

5.11 Klammersetzung mit unterschiedlichen Klammern

Erweitere das Programm Klammersetzung Rund aus Aufgabe 5.10 so, dass auch Terme mit unterschiedlichen Klammern (rund, eckig und geschwungen) auf Ihre Gültigkeit hin überprüft werden.

6 Methoden

Alle Probleme aus den Bereichen Bedingungen, einfache Programme, Schleifen, Felder, welche ein Ergebnis berechnen und dieses am Bildschirm ausgeben, können als Funktion umgeschrieben werden. Beachte dabei, dass nur die Berechnung in der zu schreibenden Methode, die Ausgabe am Bildschirm jedoch immer in der main-Methode erfolgen soll.

Term	r / f	Begründung
`[(a+b)*c+10]/d`	r	
`[(a+b]*c+10)/d`	f	`]` vor `)`
`{[(a+b)*c+10}-100]/d`	f	`}` vor `]`
`{[(a+b)*c+10}/d`	f	`]` fehlt
`{[(a+b)*c+10]-100}/d`	r	
`[(a+b)*{c-10}/10]/d`	f	richtig wenn die Reihenfolge der Klammern nicht berücksichtigt wird

Tabelle 14: Beispiele Klammersetzung2

6.1 Potenz

Erstelle eine Funktion Potenz welche als Parameter die Basis b und den Exponenten e erhält und als Rückgabe b^e zurückgibt. Basis ist ein **double**-Wert, der Exponent eine Ganzzahl.

6.2 Fehlersuche 1

```
1  import corejava.*;
2  class findestDuFehler
3  {
4    static int funktion1(int zahl1, int zahl2)
5    {
6      short differenz;
7      differenz = zahl1-zahl2;
8      return differenz;
9    }
10   static funktion3(int zahl1, int zahl2)
11   {
12     double durchschnitt;
13     durchschnitt = (zahl + zahl2)/2;
14     return durchschnitt;
15   }
16   static int funktion2(int zahl1, int zahl2)
17   {
18     int summe;
19     summe = zahl1 + zahl2;
20   }
21   static void main (String[] args)
22   {
23     zahl1 = Console.readInt("Bitte␣erste␣Zahl␣eingeben!␣");
24     zahl2 = Console.readInt("Bitte␣zweite␣Zahl␣eingeben!␣");
25     int summe;
26     int differenz2;
27     int durchschnitt;
28     int zahl3 = 10;
29     differenz2 = funktion1(zahl1,zahl3);
30     summe = funktion2(int zahl1, int zahl2);
31     durchschnitt = funktion3(zahl2);
32     summe2=funktion2(zahl2,zahl3);
33     System.out.println("Differenz:␣" + differenz2);
34     System.out.println("Summe:␣" summe);
35     System.out.println("Durchschnitt:␣" + durchschnitt);
36     System.out.println("Summe2:␣" + summe2);
```

```
37    }
38  }
```

6.3 Fehlersuche 2

```
1  public class fehlermeldungn
2  {
3    static int potenz(int basis, int exponent)
4    {
5      int ergebnis=1;
6      int i;
7      for (i=1;i<=exponent;i++)
8      {
9        ergebnis = ergebnis * basis;
10     }
11     return ergebnis;
12   }
13   static void main(String[] parameter)
14   {
15     int ergebnis;
16     double zahl1 = 2.5
17     int zahl2 = 10;
18     int[] testfeld = new int(10);
19
20     ergebnis = poten(zahl1, zahl2);
21     System.out.print(zahl1 + "␣hoch␣"+ zahl2);
22     System.out.println("␣ergibt:␣" + ergebnis);
23     for (int i = 1;i<=10;i++)
24     {
25       testfeld[i]= (int)(Math.random()*100;
26     }
27   }
28 }
```

6.4 Runden

Erstelle eine Funktion, die einen übergebenen float-Wert zu einer Ganzzahl rundet. Ab 0,5 soll aufgerundet werden.

Beispiel:

runden(-2.6) = -3

runden(2.3) = 2

runden(1.5) = 2

6.5 Runden mit Angabe der Nachkommastellen

Erstelle eine Funktion, die als ersten Parameter einen float-Wert x (zwischen 0 und 1000) und als zweiten Parameter einen positiven int-Wert n (zwischen 1 und 5) übergeben bekommt. Die Funktion soll den Wert x auf n Nachkommastellen runden.

Beispiel: runden(2.2576f, 3) ergibt 2.258f

6.6 Überprüfen ob eine Ziffer in einer Zahl vorkommt

Erstelle eine Funktion, die als Eingangsparameter eine Ganzzahl bekommt und überprüft, ob in ihr die Ziffer 7 vorkommt.

Beispiel:

ist7enthalten (-2578) = **true**

ist7enthalten (1002) = **false**

6.7 Summe von 1 bis n

Erstelle eine Methode, die als Eingangsparameter eine Ganzzahl n bekommt und als Ergebnis die Summe der Zahlen zwischen 0 und n liefert ($\sum_{i=1}^{n} i$). Ist n jedoch kleiner als 0, soll die Funktion den Wert -1 liefern.

Beispiel:

summebisn(4) = 10, summebis(-2) = -1, summebisn(100) = 5050, summebisn(0) = 0

6.8 Fakultät

Erstelle aus dem für die Aufgabe 3.41 erstellten Programm eine Methode **long** fakultaet (**int** n), welche die Fakultaet einer Zahl n berechnet. Ist n jedoch kleiner als 0, soll die Funktion den Wert -1 liefern.

6.9 Berechnung der Eulerschen Zahl

Benutze die Formel $e = \sum_{i=0}^{\infty} \frac{1}{i!}$ zur Berechnung der Eulerschen Zahl e. Man kann eine Näherng von e erreichen, wenn i nicht Werte von 0 bis ∞ durchläuft sondern nur bis n. Je grösser n ist, umso exakter ist die Näherung. Verwende zur Berechnung der Fakultät $n!$ die in der Aufgabe 6.8 erstellte Methode. Erstelle eine Methode welche als Parameter die Anzahl der durchläufe erhält und die berechnete Näherung zurückgibt. Achte auf korrekte Wahl der Datentypen!

Beispiel:

Bei 8 Schritten ergibt sich folgende Näherung für e

$\frac{1}{0!} + \frac{1}{1!} + \frac{1}{2!} + \frac{1}{3!} + \frac{1}{4!} + \frac{1}{5!} + \frac{1}{6!} + \frac{1}{7!} + \frac{1}{8!} =$

$\frac{1}{1} + \frac{1}{1} + \frac{1}{2} + \frac{1}{6} + \frac{1}{24} + \frac{1}{120} + \frac{1}{720} + \frac{1}{5040} + \frac{1}{40320} =$

$2,718278769841269841269841269841 3$

Alternative: Der Algorithmus soll abbrechen, sobald e auf 10 Stellen nach dem Komma exakt berechnet ist.

6.10 Summe von zwei Uhrzeiten

Erstelle eine Funktion, die die Summe zweier Uhrzeiten als Ergebnis liefert; Uhrzeiten werden dabei als **float**-Werte realisiert, wobei die Vorkommastellen die Stunden und die Nachkommastellen die Minuten darstellen. 22.13 steht also für 22 Uhr und 13 Minuten.

Beispiel:

uhrzeitSummieren(22.13f, 3.48f) ergibt 26.01f

6.11 Primzahlkontrolle

Erstelle analog zum Programm 3.34 Primzahlkontrolle eine Funktion, die überprüft, ob es sich bei der übergebenen Zahl (**int**) um eine Primzahl handelt oder nicht. Die Rückgabe der Funktion soll vom Datentyp **boolean** sein.

Beispiel:

istPrim(9) ergibt **false**

istPrim(11) ergibt **true**

6.12 Primzahlen erzeugen

Erstelle eine Methode, die als Eingangsparameter einen **int**-Wert n bekommt und als Rückgabe ein Feld mit n Elementen vom Datentyp **int** liefert, welches die ersten n Primzahlen (beginnend von 1) enthält. Verwende dabei die unter Übung 6.11 erstellte Methode.

Beispiel:

primGenerator(7) ergibt [1,2,3,5,7,11,13]

6.13 Primzahlen erzeugen Erathostenes

Erstelle zur Aufgabe 6.12 eine weitere Methode primGeneratorErathostenes welche Primzahlen mit Hilfe des Sieb des Erathostenes (vergleiche Aufgabe 4.9) berechnet. Stelle einen Geschwindigkeitsvergleich an.

6.14 Fibonaccizahlen

Die Fibonacci-Zahlen sind eine der bedeutendsten Zahlenfolgen. Sie fangen mit 0 und 1 an, und dann ist jede Fibonacci-Zahl gleich der Summe der beiden vorhergehenden Fibonacci-Zahlen Die ersten 15 Fibonacci-Zahlen sind: 0, 1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89, 144, 233, 377. Erstelle eine Funktion welche die n-te Fibonacci-Zahl berechnet.

Beispiel:

getFibo(9) ergibt 21

6.15 Test auf Fibonaccizahl

Erstelle eine Funktion, die testet, ob eine als Parameter übergebene natürliche Zahl eine Fibonaccizahl ist oder nicht.

Beispiel:

istFibonacci (8) ergibt **true**

6.16 Zufallszahlen erzeugen mit Rückgabe eines Feldes

Erstelle eine Funktion erzeugeZufallszahlen, die als Parameter drei int-Werte: *anzahl*, *untergrenze* und *obergrenze* übergeben bekommt und die als Ergebnis ein FelObergrenzed von *anzahl* **int**-Zahlen mit Zufallswerten im Bereich Untergrenze, liefert. Beachte, dass die Funktion ein Feld liefern soll, nicht eine einzelne Zahl.

6.17 Ausgabe eines Feldes am Bildschirm

Erstelle eine Funktion ausgabeBS, die als Parameter ein Feld **int** [] übergeben bekommt und die Elemente des Feldes untereinander am Bildschirm ausgibt.

6.18 Verschiedene mathematische Funktionen

Erstelle unten aufgezählte mathematische Funktionen, Eingangsparameter ist dabei jeweils ein Feld vom Datentyp **int**, Rückgabe jeweils eine Zahl. Achte auf korrekte Wahl des Datentyps der Rückgabe.

1. Mittelwert
2. Minimum
3. Maximum
4. Spannweite (Maximum minus Minimum). Verwende dabei die bereits erstellten Funktionen berechneMaximum und berechneMinimum.

6.19 Index des größten Elements

Schreibe eine Methode, die als Argument ein Feld von **int**-Werten bekommt und den *Index* der größten Elements zurückgibt.

Die Deklaration der Methode soll **int** maxPosition(**int** [] feld) sein.

6.20 Lottotipps

Erstelle eine Funktion, welche einen Lottotipp errechnet. Die Zufällig gewählten Zahlen liegen im Bereich von 1 bis 49 und dürfen jeweils nur einmal vorkommen. Die Kombination 1,1,3,6,8,9 ist demnach nicht gültig. Rückgabe der Funktion ist ein Feld vom Datentyp **int**.

6.21 Berechnung der Prüfziffer einer ISB-Nummer

Die ISB-Nummer (kurz: ISBN) ist eine eindeutige Nummer für Bücher. Jedes Buch kann anhand seiner ISBN eindeutig identifiziert werden. Sie besteht aus 10 Zeichen. Beispiel: 3-446-19312-8

3 Steht für das Erscheinungsland des Buchs (3=Deutschland, 0=USA ...), 446 ist die Nummer des Verlags innerhalb des Erscheinungslands, 19312 ist die Nummer des Buchs vom jeweiligen Verlag und 8 ist schlussendlich das Prüfzeichen.

In Programmen sollten Fehleingaben des Benutzers so weit möglich erkannt und gegebenenfalls nicht akzeptiert werden. Bei kryptischen Zeichenfolgen wie der ISB-Nummer kommt es häufig zu Tippfehlern. Deshalb verwendet man sogenannte Prüfzeichen. Das Prüfzeichen (meist das letzte Zeichen) errechnet sich aus den restlichen Zeichen. Hat ein Benutzer eine falsche Nummer eingegeben, so stimmt das eingegebene Prüfzeichen nicht mit dem errechneten überein und die Eingabe wird nicht akzeptiert.

Beispiel:

Der Benutzer gibt die ISB-Nummer 3-446-19312-8 ein. Die (vom Benutzer eingegebene) Prüfziffer 8 steht an der letzten Stelle. Aus den vorhergehenden Zahlen (344619312) kann die Prüfziffer errechnet werden. Das Ergebnis ist jedoch ein "X", der Benutzer hat wahrscheinlich die Nummer falsch eingegeben. Es ist sehr unwahrscheinlich jedoch nicht ausgeschlossen(!), dass die eingegebene Prüfziffer mit der berechneten Prüfziffer bei einer Fehleingabe trotzdem übereinstimmt.

Erstelle eine Funktion, welche die Prüfziffer einer ISB-Nummer eines Buchs berechnet. (Achtung: ISBN können auch Zeichen enthalten, nicht nur Zahlen.) Im Hauptprogramm soll kontrolliert werden, ob die Eingabe korrekt war oder nicht.

Die Prüfziffer (zehnte Ziffer) der ISB-Nummer berechnet sich wie folgt:

Man multipliziere die erste Ziffer mit eins, die zweite mit zwei, die dritte mit drei und so fort bis zur neunten Ziffer, die mit neun multipliziert wird. Man addiere die Produkte und teile die Summe ganzzahlig mit Rest durch 11. Der Divisionsrest ist die Prüfziffer. Falls der Rest 10 beträgt, ist die Prüfziffer ein X.

Beispiele:

1. ISBN 3-499-13599-[X] (Fräulein Smillas Gespür für Schnee)
   $3{\cdot}1 + 4{\cdot}2 + 9{\cdot}3 + 9{\cdot}4 + 1{\cdot}5 + 3{\cdot}6 + 5{\cdot}7 + 9{\cdot}8 + 9{\cdot}9 =$
   $3 + 8 + 27 + 36 + 5 + 18 + 35 + 72 + 81 = 285$
   $285{:}11 = 25$ Rest 10 Prüfziffer: X
2. ISBN 3-446-19313-[8] (Fermats letzter Satz)
   $3{\cdot}1 + 4{\cdot}2 + 4{\cdot}3 + 6{\cdot}4 + 1{\cdot}5 + 9{\cdot}6 + 3{\cdot}7 + 1{\cdot}8 + 3{\cdot}9 =$
   $3 + 8 + 12 + 24 + 5 + 54 + 21 + 8 + 27 = 162$
   $162{:}11 = 14$ Rest 8 Prüfziffer: 8
3. ISBN 1-57231-422-[2] (Hardcore Visual Basic)
   $1{\cdot}1 + 5{\cdot}2 + 7{\cdot}3 + 2{\cdot}4 + 3{\cdot}5 + 1{\cdot}6 + 4{\cdot}7 + 2{\cdot}8 + 2{\cdot}9 =$
   $1 + 10 + 21 + 8 + 15 + 6 + 28 + 16 + 18 = 123$
   $123{:}11 = 11$ Rest 2 Prüfziffer: 2

6.22 Kontrolle Steuernummer

Analog zur Kontrolle der ISBN soll ein Programm erstellt werden, welches die Eingabe einer Steuernummer auf ihre Gültigkeit hin überprüft. Das Prüfzeichen ist das letzte Zeichen der Steuernummer. Bei der Steuernummer FRS SFN 52H07 F205 C ist C das Prüfzeichen und errechnet sich wie folgt:

1. Die sieben an gerader Stelle stehende Zeichen werden mit 15 Tabelle in Zahlen umgewandelt.
2. die acht an ungerader Stelle stehende Zeichen werden mit Tabelle 16 in Zahlen umgewandelt.

A oder 0 = 0	I oder 8 = 8	Q = 16
B oder 1 = 1	J oder 9 = 9	R = 17
C oder 2 = 2	K = 10	S = 18
D oder 3 = 3	L = 11	T = 19
E oder 4 = 4	M = 12	U = 20
F oder 5 = 5	N = 13	V = 21
G oder 6 = 6	O = 14	W = 22
H oder 7 = 7	P = 15	X = 23
		Y = 24
		Z = 25

Tabelle 15: Zeichen an gerader Stelle

A oder 0 = 1	I oder 8 = 19	Q = 6
B oder 1 = 0	J oder 9 = 21	R = 8
C oder 2 = 5	K = 2	S = 12
D oder 3 = 7	L = 4	T = 14
E oder 4 = 9	M = 18	U = 16
F oder 5 = 13	N = 20	V = 10
G oder 6 = 15	O = 11	W = 22
H oder 7 = 17	P = 3	X = 25
		Y = 24
		Z = 23

Tabelle 16: Zeichen an unterader Stelle

6.23 Berechnen Steuernummer

Erstelle ein Programm, welches den Steuerkodex einer natürlichen Person berechnet. Die Vorgehensweise beim Berechnen der Steuernummer ist folgende:

1. Nachname: man nimmt den ersten, den zweiten und den dritten Konsonanten des Nachnamens.

   Ausnahmen: Wenn der Nachname nur zwei Konsonanten hat, werden diese beiden Konsonanten und der erste Vokal verwendet. Sollte der Nachname nur einen einzigen Konsonanten haben, nimmt man diesen einen Konsonanten und die ersten zwei Vokale. Wenn der Nachname nur aus einem Konsonanten und einen Vokal besteht nimmt man den Konsonanten, den Vokal und an dritter Stelle ein X. Fragt mich nicht, was geschieht wenn ein Nachname nur aus Vokalen besteht.

   Nachnamen, welche aus mehreren Worten bestehen werden als einziges Wort gezählt. Von Pergine wird also als VONPERGINE interpretiert.

0 = A	8 = I	17 = R
1 = B	9 = J	18 = S
2 = C	10 = K	19 = T
3 = D	11 = L	20 = U
4 = E	12= M	21 = V
5 = F	13= N	22 = W
6 = G	14= O	23 = X
7 = H	15= P	24 = Y
	16= Q	25 = Z

Tabelle 17: Umwandlung Zahl in Kontrollzeichen

2. Vorname: erster, dritter und vierter Konsonant des Vornamens.

   Ausnahmen: Wenn der Vorname nur drei Konsonanten hat, werden alle drei Konsonanten verwendet. Wenn weniger als 3 Konsonanten vorhanden sind, gelten dieselben Regeln wie beim Nachnamen!

3. Geburtsjahr: die letzten beiden Zahlen des Jahres.

4. Geburtsmonat:

   Januar = A, Februar = B, März = C, April = D, Mai = E, Juni = H, Juli = L, August = M, September = P, Oktober = R, November = S, Dezember = T.

5. Geburtstag:

   Männer: Geburtstag wird unverändert übernommen, die Tage von 1 bis 9 werden zweistellig mit führender 0 angegeben.

   Frauen: Zum Geburtstag wird die Zahl 40 hinzugezählt.

6. Geburtsort: Der Geburtsort wird mit 4 Zeichen angegeben (ein Buchstabe gefolgt von 3 Zahlen). Für jeden Geburtsort gibt es einen eindeutigen Kodex. So hat Meran z. B den Kodex F132. Da wir nicht alle Gemeinden erfassen können beschränken wir uns auf 2 oder 3 verschiedene.

7. Kontrollzeichen: wie in der Übung 6.22 angegeben, wird zusätzlich ein Kontrollzeichen angefügt. Verwende zur Berechnung die bereits erstellte Funktion.

6.24 Umrechnen zwischen verschiedenen Zahlensystemen

Erstelle ein Programm, welches eine eingegebene Zahl in ein anderes Zahlensystem umrechnet. Die Eingaben sollen in einem Zahlensystem mit Basis kleiner oder gleich 16 (Hexadezimalsystem) erfolgen. Die Ausgaben ebenso.

Beispiel:

```
Zahlensysteme
*************

von Zahlensystem: 16
nach Zahlensystem: 10
```

```
Umzurechnende Zahl: 16a
16a(16) = 363(10)

Umzurechnende Zahl: 11
11(16) = 17(10)

Umzurechnende Zahl: 1g
1g ist keine Zahl des 16ersystems

Umzurechnende Zahl: 0
bye.
```

Folgende Punkte soll dein Programm erfüllen.

- Bei Fehleingabe soll die Eingabe wiederholt werden.
- Beendet wird das Programm mit Eingabe von 0 als umzurechnende Zahl.
- Schreibe eine Methode zur Umrechnung
  String umrechnung(String zahl, **int** vonsys, **int** nachsys)
- und eine Methode zur Kontrolle
  boolean istImZahlensystem(String zahl, **int** system).

Die umzurechnende Zahl beziehungsweise das Ergebnis müssen vom Datentyp String sein, da in Zahlensystemen mit Basis grösser als 10 Buchstaben vorkommen können.

6.25 Merge

Schreib eine Funktion merge (englisch: mischen), die zwei sortierte Felder mit Elementen vom Typ int zusammenfügt, so dass wieder ein sortiertes Array entsteht.

Die Deklaration der Methode muss folgende sein:
int [] merge(**int** [] feldA, **int** [] feldB)

Beispiel:

Die Felder

x:

1	2	7	11	18

und

y:

2	3	4	6	22	22

werden mit der Funktion merge zu

merge(x,y):

1	2	2	3	4	6	7	11	18	22	22

7 nicht Klassifiziert

7.1 Normalverteilte Zufallsvariable

Wenn eine hinreichend große Anzahl ($n >= 12$) von gleichverteilten Zufallsvariablen summiert erhält man eine normalverteilte Zufallsvariable.

Simuliere diese Aussage indem du ein Programm schreibst in welchem du n gleichverteilte Zufallszahlen wie sie von Math.random() geliefert werden aufsummierst und die so erhaltene Zufallsvariable grafisch als Histogramm darstellst und vergleichst, ob das Histogramm mit der aus dem Mathematikunterricht bekannten gaußschen Glockenkurve übereinstimmt.

7.2 Scheck: Zahl in Worte

Ein Unternehmen beschäftigt zahlreiche Vertreter. Viele dieser möchten ihre Provision aus unterschiedlichen Gründen nicht per Banküberweisung erhalten sondern am als Scheck. Da die Buchhaltungsabteilung bereits hoffnungslos überarbeitet ist wünscht diese, dass die aus den Verkaufszahlen automatisch errechnete Provisionssumme direkt auf Scheckformulare gedruckt wird. Der Beauftragte Software-Entwickler schafft es jedoch nicht, den in Zahlen vorliegenden Betrag in Worte umzuwandeln. Er bittet dich um Hilfe: „Erstelle mir bitte eine Funktion (Methode) welche als Eingangsparameter einen Ganzzahligen Betrag erhält und als Rückgabewert einen Text mit dem angegebenen Betrag erhält.“ Beispiel 100210 –¿ einhunderttausendzweihundertundzehn. Da die Schecks in € ausgestellt werden genügt es wenn Beträge zwischen 1 und 999.999 berücksichtigt werden. (Bei Zahlen ¿ 1.000.000 müssten etliche Sonderfälle betrachtet werden.)

Bemerkung: rekursiv ist dieses Problem mit relativ wenigen Programmzeilen gelöst.

7.3 Sortieralgorithmen

7.3.1 Bubblesort

Schreibe eine Methode welche ein übergebenes Integer-Feld beliebiger Länge mit dem Sortieralgorithmus „Bubble-Sort“ sortiert. Die Methode soll wie folgt deklariert werden: **int** [] bubbleSort(**int** [] unsortiertesFeld)

Ein Feld wird einmal vollständig durchlaufen. Dabei werden die jeweiligen Nachbarelemente miteinander verglichen und ggf. ausgetauscht. Am Ende befindet sich das größte Element am Ende des Feldes. Dieser Schritt wird nun mit dem kleineren Teilfeld (Feld ohne das letzte Element) wiederholt und wiederholt und . . . Und irgendwann sind wir fertig und die Elemente sind sortiert.

In einer Wassersäule befinden sich unterschiedlich große Luftblasen. Die unterste Luftblase steigt nun langsam auf. Während sie an kleineren problemlos vorbeikommt, wird sie durch größere gestoppt. Durch den Zusammenprall setzt sich die größere Luftblase in Bewegung, bis auch diese gestoppt wird oder am oberen Säulenende ankommt. Nun setzt sich abermals die unterste Luftblase in Bewegung bis . . . Und wenn die Wassersäule eine endliche Anzahl Luftblasen enthält, dann kann irgendwann keine Luftblase mehr aufsteigen, da sich über ihr eine noch größere befindet. Die Luftblasen sind dann der größe nach von unten (klein) nach oben (groß) sortiert.

Eine Seifenblase kann genau zwei Elemente in sich einschließen. Diese Seifenblase steigt nun langsam auf. Das schwerere der beiden Elemente in der Blase kann nicht nach oben getragen werden und fällt unten raus, während oben ein neues Element hineinkommt. Dadurch trägt die Seifenblase das leichteste Element bis ganz nach oben. Hier zerplatzt die Seifenblase und unten entsteht eine neue.

Aufgrund der Vergleiche benachbarter Elemente wird dieses Verfahren auch als „Sortierung durch Nachbarvergleiche“ bezeichnet.

Erklärung entnommen aus http://www.sortieralgorithmen.de.

7.3.2 Mergesort

Schreibe eine Methode welche ein übergebenes Integer-Feld beliebiger Länge mit dem Sortieralgorithmus „Merge-Sort“ sortiert. Die Methode soll wie folgt deklariert werden: **int** [] mergeSort(**int** [] unsortiertesFeld)

Ein Feld wird in zwei Teilfelder aufgeteilt, die dann rekursiv sortiert werden. Anschließend werden diese sortierten Teilfelder wieder zu einem Feld zusammengefügt. Dabei macht man sich zu nutze, dass die beiden Teilfelder bereits sortiert sind.

Erklärung entnommen aus http://www.sortieralgorithmen.de.

Alternative: sowohl die rekursive als auch die iterative Lösung. Führe einen Geschwindigkeitsvergleich beider Verfeahren durch.

7.3.3 Quicksort

Ein Feld wird in zwei (in der Regel unterschiedlich große) Teilfelder aufgeteilt, die Elemente werden dabei so vertauscht, daß alle Elemente des linken Teilfeldes kleiner (oder gleich) den Elementen des rechten Teilfeldes sind. Die einzelnen Teilfelder werden dann wieder sortiert. . . Und irgendwann sind wir fertig und das gesamte Feld liegt sortiert vor.

Die Aufteilung eines Feldes in zwei Teilfelder geschieht aufgrund von Vergleichen mit einem speziellen (Am Anfang der Teilung gewählten) Pivotelement. Deshalb wird dieses Verfahren auch als „Sortierung durch Pivotisierung“ oder „Sortierung durch Partitionierung“ bezeichnet.

Erklärung entnommen aus http://www.sortieralgorithmen.de

7.4 GGT mit dem Euklidischen Algorithmus

Erstelle ein Programm welches mit Hilfe des klassischen und binären euklidischen Algorithmus den GGT zweier Zahlen berechnet.

7.4.1 Beschreibung des Algorithmus

aus Wikipedia, der freien Enzyklopädie
http://de.wikipedia.org/wiki/Euklidischer_Algorithmus

Der Euklidische Algorithmus ist ein Verfahren zur Bestimmung des größten gemeinsamen Teilers (ggT) zweier natürlicher Zahlen a und b. Es ist der älteste bekannte Algorithmus der Welt, benannt nach dem griechischen Mathematiker Euklid, der ihn um 300 vor Christus in seinem Werk „Die Elemente“ angegeben hat. Das Verfahren war jedoch schon früher bekannt. Euklid nannte es antenaresis.

7.4.2 Der klassische Algorithmus

Das Prinzip des Euklidischen Algorithmus wird auch gegenseitige Wechselwegnahme genannt. Eingangsgrößen sind zwei natürliche Zahlen a und b. Bei der Berechnung verfährt man nach Euklid wie folgt:

1. setze $m = a$; $n = b$
2. ist $m < n$, so vertausche m und n
3. berechne $r = m - n$
4. setze $m = n$; $n = r$
5. ist $r = 0$ fahre fort mit Schritt 2

Nach Ablauf des Verfahrens hat man mit m den ggT von a und b gefunden.

Ist die Differenz von a und b sehr groß, sind unter Umständen sehr viele Subtraktionsschritte notwendig. Oftmals werden die Schritte 2 und 3 deshalb dadurch ersetzt, dass man statt der Subtraktion r als Divisionsrest von m und n nimmt. Ein weiterer Vorteil dieser Variante ist, dass man sie auf andere algebraische Strukturen (zum Beispiel Polynomringe, siehe Ringtheorie) übertragen kann, in denen der klassische Algorithmus nicht funktioniert.

Mit dem Euklidischen Algorithmus kann man den ggT mit verhältnismäßig geringem Aufwand (im Vergleich zur Primfaktorzerlegung) berechnen. Bei der Laufzeitanalyse stellt sich interessanterweise heraus, dass der schlimmste Eingabefall zwei aufeinander folgende Fibonacci-Zahlen sind.

7.4.3 Der binäre Euklidische Algorithmus

Ein Problem bei der Umsetzung des Euklidischen Algorithmus auf Computer ist Division, die unter Umständen einen hohen Rechenaufwand bedeutet. Hier ist deshalb der binäre euklidische Algorithmus besonders geeignet. Er verwendet nur Subtraktion und die im Binärsystem besonders einfach durchzuführende Division durch 2.

1. setze $m = a$; $n = b$
2. dividiere m und n durch 2 solange, bis eine der beiden Zahlen ungerade ist. Die Zahl der notwendigen Divisionsschritte sei k
3. dividiere m durch 2, bis m ungerade ist
4. ist $m < n$, so vertausche diese Zahlen
5. setze $m = m - n$
6. ist $m = 0$, dann fahre fort mit Schritt 3.

Nach Ablauf erhält man $ggT(a, b) = n * 2k$.

Hinter dem binären euklidischen Algorithmus steckt die Tatsache, dass 2 kein Faktor des ggT zweier Zahlen sein kann, wenn mindestens eine der beiden ungerade ist.

Aus einer geraden Zahl kann man also so lange 2 ausdividieren, bis diese ungerade wird. Dies geschieht in Schritt 3. Wenn im Schritt 5 von einer ungeraden Zahl eine ungerade Zahl abgezogen wird - was immer der Fall ist -, ist das Ergebnis eine gerade Zahl, aus der man nun wieder 2 ausdividieren kann. Die Bitlänge der Restzahlen verringert sich so kontinuierlich.

Das einzige Problem ergibt sich bei der Eingabe zweier gerader Zahlen. Hier muss man im Voraus entsprechend oft 2 ausdivieren (Schritt 2). Die Zahl der Divisionen muss man sich merken, da diese nach Beendigung des Algorithmus wieder rückgängig gemacht werden müssen.

7.5 Nim Spiel

Gegeben ist ein Stapel mit s Streichhölzern, von dem 2 Spieler abwechselnd mindestens 1 und höchstens n Streichhölzer wegnehmen. Wer das letzte Streichholz nimmt, hat gewonnen.

Erstelle ein Programm, in dem der Computer als Spielgegner auftritt. Die Anfangsgröße des Stapels und die maximal in einem Zug zu nehmenden Streichhölzer werden vom Benutzer beim Programmstart eingegeben. Bei jedem Zug gibt das Programm aus, wie viele Streichhölzer noch am Stapel sind (evtl. grafischer Ausgabe mit Sternchen) und Bei jedem Zug gibt das Programm den aktuellen Stapel aus (evtl. grafische Ausgabe als Histogramm mit Sternchen), fragt wieviele Streichhölzer du nehmen möchtest, gibt dann den Stapel erneut aus, entfernt selbst einige Streichhölzer und wiederholt dieses Prozedere bis ein Spieler gewonnen hat.

Gewinnstrategie:

Setze zug gleich s modulo (n+1)

Wenn zug ¿ 0 gilt, dann nimm zug Streichhölzer vom Stapel. Ansonsten besteht sowieso keine Chance, in eine Gewinnposition zu kommen. Nimm deshalb eine beliebige Menge von Streichhölzern (zwischen 1 und n).

Für genauere Informationen siehe http://www.mathematische-basteleien.de/nimspiel.html.

7.6 Wochentag feststellen

Erstelle ein Programm, das vom Benutzer die Eingabe von Tag, Monat und Jahr verlangt und den Wochentag als Text ausgibt.

Hinweis: Der 15.10.1583 war ein Sonntag. Berechne wieviele Tage zwischen dem 15.10.1583 und dem Eingabedatum liegen. Diese Zahl Modulo 7 ergibt den Wochentag. Siehe Aufgaben zum Schaltjahr.

7.7 Bremsweg berechnen

Unter Bremsweg versteht man die Strecke, die ein Fahrzeug vom Beginn der Bremsung bis zum Stillstand zurücklegt. Entscheidend für die Länge des Bremsweges ist die gefahrene Geschwindigkeit und die Verzögerung.

Nach Eingabe der Geschwindigkeit soll der Bremsweg ausgegeben werden. Benutze

für die Bremsverzögerung die Fahrschul-Formel für den normalen Bremsweg (Bremsverzögerung 3,86 m/s^2).

$$s = \frac{v^2}{2a}$$

s = Bremsweg in m

v = Geschwindigkeit in m/s

a = Bremsverzögerung in m/s^2

7.8 ***** TODO: Cosinus berechnen

Der Cosinus kann durch eine Reihenentwicklung approximiert werden. Die Reihenentwicklung schaut so aus:

GIOVANNI BILD!!!!!

Schreibe ein Programm, das den Cosinus mit dieser Approximation annähert. Die Approximation soll abbrechen, sobald der Betrag einer Partialsumme einen Schwellenwert unterschritten hat.

7.9 ***** TODO: Binäre Suche

Die binäre Suche ist ein Algorithmus, der in einer Liste (bzw. einem Array) recht schnell ein gesuchtes Element findet bzw. eine zuverlässige Aussage über das Fehlen dieses Elementes liefert. Voraussetzung ist, dass die Elemente der Liste in einer dem Suchbegriff entsprechenden Weise sortiert sind.

Erstelle ein Programm (eine Methode) welches die binäre Suche in einem Feld implementiert.

8 Rekursion

8.1 Fakultät rekursiv

Erstelle eine rekursive Funktion **long** fakrec(**int** n) welche die Fakultät berechnet. Es gilt folgende rekursive Definition der Fakultät:

$$n! = 1 \, \forall n = 0$$

$$n! = (n-1)! \, \forall n > 0$$

8.2 Quersumme rekursiv

Schreibe eine *rekursive* Methode **int** quer(**int** zahl), die die Quersumme einer ganzen Zahl berechnet. Die Quersumme einer Zahl ist die Summe der Dezimalziffern der Zahl. Beispielweise hat die Zahl -123456 die Quersumme 21.

8.3 GGT rekursiv

Schreibe eine Methode **int** ggt(**int** x, **int** y), die zu zwei positiven ganzen Zahlen den größten gemeinsamen Teiler berechnet.

Verwende dabei die folgenden Gleichungen:

$$\begin{aligned} ggt(x,x) &= x & x &> 0 \\ ggt(x,y) &= ggt(x-y,y) & x > y &> 0 \\ ggt(x,y) &= ggt(y,x) & x, y &> 0 \end{aligned}$$

8.4 Variationen

Alle Variationen von n Elementen eines Felds sollen am Bildschirm ausgegeben werden. Beispiel: aus dem Feld

1	2	3

resultieren folgende Variationen:

1	2	3
1	3	2
2	1	3
2	3	1
3	1	2
3	2	1

8.5 Türme von Hanoi

aus: http://de.wikipedia.org/wiki/T\protect\unhbox\voidb@x\bgroup\U@D1ex{\setbox\z@\hbox{\char127}\dimen@-.45ex\advance\dimen@\ht\z@}\accent127\fontdimen5\font\U@Du\egrouprme_von_Hanoi

Die Türme von Hanoi sind ein mathematisches Knobel- und Geduldsspiel.

Es besteht aus drei Feldern, auf die Scheiben verschiedener Größe gelegt werden können. Zu Beginn sind alle Scheiben auf einem Feld, der Größe nach geordnet, mit der größten Scheibe unten und der kleinsten oben. Bei jedem Zug darf die oberste Scheibe eines beliebigen Feldes auf eines der beiden anderen Felder gelegt werden, vorausgesetzt, dort liegt nicht schon eine kleinere Scheibe. Ziel des Spiel ist es, den kompletten Scheiben-Stapel auf ein anderes Feld zu versetzen.

```
       =                |                |
      ===               |                |
     =====              |                |
---------------   ---------------   ---------------
   Startfeld                          Zielfeld
```

Vermutlich wurde das Spiel 1883 vom französischen Mathematiker Edouard Lucas erfunden. Er dachte sich dazu die Geschichte aus, dass indische Mönche im großen Tempel zu Benares, im Mittelpunkt der Welt, einen Turm aus 64 goldenen Scheiben versetzen müssten, und wenn ihnen das gelungen sei, wäre das Ende der Welt gekommen.

Die minimale Anzahl von Zügen für einen Stapel aus n Scheiben beträgt 2n-1, bei einem Turm von 8 Scheiben (die gängigste Variante) also 255 Züge. Für den Stapel aus 64 Scheiben würden 18.446.744.073.709.551.615, also mehr als 18 Trillionen Züge benötigt. Würde man jede Sekunde eine Scheibe bewegen, bräuchte man dafür über 584 Milliarden Jahre.

Obwohl es auf den ersten Blick kompliziert klingt, gibt es für das Spiel einen ganz einfachen rekursiven Algorithmus. Da sich ein Computerprogramm zur Lösung des Spiels mit wenigen Zeilen schreiben lässt, ist Türme von Hanoi ein klassisches Beispiel für rekursive Programmierung.

n Scheiben vom Stapel a auf den Stapel c zu verschieben entspricht folgender Vorgehensweise:

- $n-1$ Scheiben vom Stapel a auf den Stapel b
- die noch am Stapel a verbleibende Scheibe auf den Stapel c
- die $n-1$ Scheibem vom Stapel b auf den Stapel c

Das Verschieben von $n-1$ Scheiben wird wiederum mit demselben Algorithmus gelöst. Erst das Verschieben einer einzelnen Scheibe kann unmittelbar durchgeführt werden.

Erstelle ein Programm, welches die einzelnen Züge in folgender Form ausgibt.

a $\rightarrow$ b

a $\rightarrow$ c

b $\rightarrow$ c

...

Dies bedeutet: Verschiebe die oberste Scheibe des Stapels a auf den Stapel b, die oberste Scheibe vom Stapel a auf den Stapel c und die oberste Scheibe vom Stapel b auf den Stapel c...

9 Lizenz