Übungen Kapitel 3

3.1 (*) Spaghetti-Code *(GOTO Kontrollstrukturen)*

Gegeben ist folgendes Pseudo Pascal-Codefragment:

```
   GOTO 3;
1: IF x = 0 THEN GOTO 9
            ELSE GOTO 5
5: IF x > 2005 THEN GOTO 6
                ELSE GOTO 4;
9: WRITELN( x );            // Ausgabe von x auf der Console
   GOTO 7;
3: READLN( x );              // Einlesen eines Wertes von der
                             // Console und speichern des
                             // Wertes in der Variablen x
   GOTO 1;
6: x := SQRT( x );           // SQRT(x) bedeutet Wurzel von x
8: x := x * x;
   GOTO 9;
4: x := x * (x + x);
   GOTO 8;
7: ...
```

Schreiben Sie das folgende Pascal-Codefragment in optimierten Java-Code um, wobei zu berücksichtigen ist, dass Java kein GOTO-Statement wie in Pascal kennt.

Lösungshinweis: Die erste Spalte zeigt so genannte Sprungmarken: d.h. z.B. die nächste Zeile, die nach der ersten Zeile abgearbeitet wird, ist die Zeile mit Marke 3 (READLN(x)). Für SQRT verwenden Sie wieder `Math.sqrt`.

3.2 `()` `if` Fragen *(if-Anweisung)*

Ermitteln Sie die Ausgabe des Programms für die folgenden Code Fragmente, falls `x` den Wert 9 und `y` den Wert 11 sowie falls `x` den Wert 11 und `y` den Wert 9 haben. Beachten Sie, dass ein Java Compiler immer ein `else` solange mit dem vorigen `if` assoziiert bis `{ }` gesetzt sind. Auf Einrückungen ist bewusst verzichtet worden:

a)
```
if (x < 10)
  if (y > 10)
  System.out.println("*****");
  else
  System.out.println("######");
  System.out.println("$$$$$$");
```

b)
```
if (x < 10) {
  if (y > 10)
  System.out.println("*****");
  }
```

```
else {
System.out.println("######");
System.out.println("$$$$$$");
}
```

3.3 () Zusatzaufgabe: "Dangling-Else" Problem (*Verschachtelte if-else-Anweisung)*

Ändern Sie den gegebenen Programmcode dahingehend ab, dass jeweils die Ausgabe erzeugt wird, die in den einzelnen Unteraufgaben vorgegeben ist. Erlaubte Änderungen am Code: Einrückungen und Klammern. Dabei dienen die Einrückungen im Quellcode nur der Übersichtlichkeit und werden vom Compiler ignoriert:

```
if ( y == 8 )
if ( x == 5 )
System.out.println( "@@@@@");
else
System.out.println( "#####");
System.out.println( "$$$$$");
System.out.println( "&&&&&");
```

a) Für x = 5 und y = 8 soll die folgende Ausgabe erzeugt werden:

```
@@@@@
$$$$$
&&&&&
```

b) Für x = 5 und y = 8 soll die folgende Ausgabe erzeugt werden:

```
@@@@@
```

c) Für x = 5 und y = 8 soll die folgende Ausgabe erzeugt werden:

```
@@@@@
&&&&&
```

d) Für x = 5 und y = 7 soll die folgende Ausgabe erzeugt werden: (Hinweis: die letzten drei Anweisungen nach der else-Anweisung sind als ein Block anzusehen!)

```
#####
$$$$$
&&&&&
```

3.4 (*) Mäxchen *(einfacher Algorithmus, if-else Anweisung)*

Mäxchen ist ein einfaches Würfelspiel, dessen Spielregeln hier ohne Bedeutung sind. Ein Spieler wirft zwei Würfel, wobei sich der Wert des Wurfs aus der Augenzahl des Würfels wie folgt ergibt:

- Der Wurf 1, 2 heißt Mäxchen und ist 1000 Punkte wert.
- Ein Wurf mit zwei gleichen Augenzahlen wird als Pasch bezeichnet und ist 100 · Augenzahl an Punkten wert. Der Wurf 4, 4 hat beispielsweise den Wert 400.
- Ansonsten ist der Wert 10· (höhere Augenzahl)+(niedrige Augenzahl). Der Wurf 3, 5 hat beispielsweise den Wert 53.

Schreiben Sie ein Programm `Maexchen`, das die Augenzahl von zwei Würfeln vom Bediener einliest und den Wert des Wurfes ausgibt. Gehen Sie davon aus, dass der Bediener immer eine korrekte Zahl von 1 bis 6 eingibt.

Beispiel:

```
>Bitte 1. Würfelzahl eingeben
2
Bitte 2. Würfelzahl eingeben
1
1000

>Bitte 1. Würfelzahl eingeben
4
Bitte 2. Würfelzahl eingeben
3
43

>Bitte 1. Würfelzahl eingeben
3
Bitte 2. Würfelzahl eingeben
3
300
```

3.5 (*) Schleifenumformungen *(while, for, do-while Schleife)*

Implementieren Sie eine Klasse `DreierSumme`, die die Summe aller positiven und durch 3 teilbaren natürlichen Zahlen bis zu einer Zahl `n` berechnet. Dabei gelten folgende Regeln:

- Lesen Sie `n` durch den Bediener ein.
- Wenn `n` kleiner als 3 ist wird als Summe 0 ausgegeben.
- Wenn n nicht durch 3 teilbar ist, wird die Summe nur bis zur nächst kleineren durch 3 teilbaren Zahl berechnet. Beispiel: Wenn n = 14 wird als Ergebnis die Summe 3+6+…+12 berechnet.

  Lassen Sie sich nicht nur die Summe ausgeben, sondern auch die Zahlen, die wirklich für die Addition verwendet werden.

  Das Programmausgabe sieht beispielhaft so aus:

```
>n eingeben:
12
3  + 6 + 9 + 12
summe = 30

>n eingeben:
2
summe = 0

>n eingeben:
16
3 + 6 + 9 + 12 + 15
summe = 45
```

Jetzt zur eigentlichen Aufgabe:

a. Implementieren Sie das Programm durch Verwendung einer `for`-Schleife.
b. Ändern Sie das Programm in a) so um, dass Sie eine `while` Schleife verwenden und die Ausgaben identisch sind.
c. Ändern Sie das Programm in a) so um, dass Sie eine `do-while` Schleife verwenden und die Ausgaben identisch sind.

Testen Sie insbesondere ihre Schleifen für n mit den Werten 12, 2, 16.

3.6 (*) Fibonacci-Reihe *(einfacher Algorithmus, Schleife)*

Die Fibonacci Reihe 0, 1, 1, 2, 3, 5, 8, 13, 21, ... startet mit $f_0 = 0$ und $f_1 = 1$ und eine Fibonacci Zahl $f_i := f_{i-1} + f_{i-2}$, wobei i = 2, 3, ,...n. Schreiben Sie ein Programm das eine beliebige Fibonacci Zahl n berechnet. Verwenden Sie ausschließlich die Ihnen bis jetzt bekannten Java Konstrukte. Testen Sie ihr Programm für den Index n = 50 und n= 95. (Lösungshinweis f_{50} = 12586269025. Achtung: Das erste Element in der Reihe wird mit f_0 (also Index 0) definiert.):

```
> Bitte Index n eingeben
0
Fibonacci Zahl: f0 = 0

> Bitte Index n eingeben
50
Fibonacci Zahl: f50 = 12586269025
```

3.7 () Zweckmässige Kontrollstrukturen (Papieraufgabe)

Das folgende Java-Programm ist zwar syntaktisch korrekt, die Kontrollstrukturen wurden jedoch unzweckmäßig verwendet. Verbessern Sie das Programm durch Verwendung der korrekten Kontrollstrukturen:

```
public static void main(String[] args)  {
              …
     boolean bedingung   =    sc.nextBoolean();
     int wert            =    sc.nextInt();

     int i;
     i = 1;
     while(i < 10){
       System.out.println(i);
       i = i + 1;
     }

    while(bedingung == true){
        System.out.println("Bedingung ist wahr");
        break; // Beendet while-Schleife
    }

    if(wert == 0){
        System.out.println("Wert ist 0");
    }
    else if(wert == 1){
        System.out.println("Wert ist 1");
    }
    else if(wert == 2){
        System.out.println("Wert ist 2");
    }
    else{
       System.out.println("Wert ist weder 0, noch 1, noch 2");
    }
}
```

3.8 () Fehler finden (Papieraufgabe) *(Kontrollstrukturen)*

Identifizieren und korrigieren Sie Fehler in folgenden Code Fragmenten. (Gehen Sie bei den Beispielen davon aus, dass die verwendeten Variablen vorher bereits deklariert sind!):

a.
```
while (c < = 5) {
    product *= c;
    ++c;
```

b)
```
if (gender == 1)
```

```
    System.out.println("Woman");
 else;
    System.out.println("Man");
```

```
c). if (a = 6)
   for(int b == 0; b < 20, b++)
        a++;
```

```
d)  while (z >=0 )
   sum += z;
```

```
e)  if  (age >= 65 );
            System.out.println("Age greater than
            or equal to 65");
    else
             System.out.println("Age is less  than
             65");
```

```
f) int x =1, total;
       while ( x <= 10 ) {
              total += x;
                  ++x;
              }
```

```
g) While ( x <= 100 )
              total += x;
              ++x;
```

```
h)  while ( y  > 0 ) {
              System.out.println( y );
              ++y;
```

```
i) x = 1;

   while( x <= 10 );
        x++;
```

```
j) for ( y = .1; y != 1.0; y+= .1 )
        System.out.println( y );
```

```
k) switch ( n ) {
     case 1:
           System.out.println("The number is 1" );
     case 2:
           System.out.println("The number is 2" );
           break;
     default:
          System.out.println("The number is not 1 or 2" );
               break;
```

```
}
```

l) Der folgende Code soll die Werte 1 bis 10 ausgeben:

```
n=1;

while ( n < 10 )
    System.out.println( n++ );
```

k)
```
For ( x = 100, x >=1, x++)
   System.out.println(x);
```

l) Folgender Code gibt aus, ob `value` gerade oder ungerade ist:

```
switch( value %2) {

 case 0:
     System.out.println("Even integer");

 case 1:
     System.out.println("Odd integer");
```

m) Folgender Code gibt die ungeraden Zahlen von 1 bis 19 aus

```
for( (x = 19; x >=1; x +=2)
   System.out.println(x);
```

n) Folgender Code gibt die geraden Zahlen von 2 bis 100 aus

```
counter = 2;

do {
   System.out.println(counter);
   counter += 2;
} While (counter < 100);
```

3.9 (*) Vokale und sonstige Zeichen *(switch-Anweisung)*

Schreiben Sie ein Programm `Vokale`, das die Vokale und sonstigen Zeichen in einem Satz zählt. Dazu lesen Sie einen Satz über die Console ein. Das Programm gibt dann die Anzahl der a's, e's, i's usw. sowie die Anzahl der Konsonanten und Sonderzeichen aus. Ein Leerzeichen wird dabei als Sonderzeichen gezählt. **Dabei wird nicht zwischen Groß- und Kleinschreibweise der Vokale unterschieden.** Außerdem wird das Programm solange weiter fortgesetzt, bis der Bediener bei Beenden `j` eintippt. Verwenden Sie zur Lösung eine `switch` Anweisung.

Ein Beispieldialog **eines** Programmlaufs sieht wie folgt aus:

```
> Bitte Satz eingeben
Du hast zwei Augen im Gesicht.
a:2
e:3
i:3
o:0
u:2
Kons. u. Sonderzeichen:20
Beenden (j oder n)
n
Bitte Satz eingeben
Ist Java cool?
a:2
e:0
i:1
o:2
u:0
Kons. u. Sonderzeichen:9
Beenden (j oder n)
n
Bitte Satz eingeben
Das Essen schmeckt gut!!
a:1
e:3
i:0
o:0
u:1
Kons. u. Sonderzeichen:19
Beenden (j oder n)
j
```

Lösunghinweis:

- **Anzahl der Zeichen in einem Satz ist `satz.length()`**
- **Auf einen einzelnen Buchstaben eines Strings greifen Sie mit `charAt` zu:**

```
String satz;
char c;
```

Erster Buchstabe im Satz:

```
c = satz.charAt(0);
```

Zweiter Buchstabe:

```
c = satz.charAt(1);
```

Letzter Buchstabe im `satz`:

```
c = satz.charAt(satz.length()-1);
```

3.10 () Ausgabe *(while, inkrement, operatoren)*

a) Welche Ausgabe erzeugen die folgenden Programme?:

```
public class Mystery {
    public static void main ( String args[ ] ){
          int y, x = 1, total = 0;
          while ( x <= 10 ) {
                y = x * x ;
                System.out.println( y );
                total += y;
                ++x;
          }
    }
}
```

b)
```
public class Mystery2 {
    public static void main ( String args[ ] ){
          int count = 1;
          while ( count <= 10 ) {
              System.out.println(
              count %2 == 1 ? "*****" : "++++++");
              ++count;
          }
    }
}
```

c)
```
public class Mystery3 {
    public static void main ( String args[ ] ){
          int row = 10, column;
          while ( row  >= 1 ) {
            column = 1;
            while ( column <= 10 ) {
              System.out.print(  row % 2 == 1 ? "<" : ">");
              ++column;
            }
            -- row;
          }
          System.out.println( );
    }
}
```

3.11 (*) Minimum *(einfacher Algorithmus, Kontrollstrukturen)*

Schreiben Sie ein Programm, das das Minimum einer Menge von ganzen Zahlen berechnet und diese durch den Bediener einliest. Lesen Sie als erstes ein, wie viel Zahlen eingelesen werden sollen.

```
Anzahl der Zahlen
5
Bitte Zahl eingeben
23
Bitte Zahl eingeben
345
Bitte Zahl eingeben
66
Bitte Zahl eingeben
-4
Bitte Zahl eingeben
0
Minimum: -4
```

3.12 (*) Sterne *(for-Schleifen)*

Schreiben Sie ein Programm, dass die folgenden Sternmuster untereinander ausgibt. Benutzen Sie **for**-Schleifen. Alle Sterne (*) sollen jeweils durch eine einzelne Anweisung der Form **`System.out.print( `*` );`** erzeugt werden . **`System.out.print( );`** verwenden Sie für den Zeilenumbruch. **`System.out.print(``);`** soll benutzt werden, um die Leerzeichen in den letzten beiden Mustern zu erhalten.

a)
```
*
**
***
****
*****
******
*******
********
*********
**********
```

b)
```
**********
*********
********
*******
******
*****
****
***
**
*
```

c)
```
**********
 *********
  ********
   *******
    ******
     *****
      ****
       ***
        **
         *
```

d)
```
         *
        **
       ***
      ****
     *****
    ******
   *******
  ********
 *********
**********
```

3.13 (*) Zusatzaufgabe: Pi *(einfacher Algorithmus, for-Schleife)*

Die Zahl Pi definiert sich durch folgende (unendliche) Reihe:

$$\pi = \quad 4 - \frac{4}{3} \quad + \frac{4}{5} \quad - \frac{4}{7} \quad + \frac{4}{9} \quad - \frac{4}{11} + \ldots$$

Schreiben Sie ein Programm, das die Zahl π approximiert, in dem erst ein Term, dann zwei, dann drei usw. berechnet werden und lassen Sie sich jeweils den aktuellen Wert ausgeben. Wieviele Terme müssen Sie berechnen, um das erste Mal 3.14? 3.141? 3.1415? 3.14159? zu erhalten? Für die Ausgabe verwenden Sie `System.out`.

a. Entwickeln Sie das Programm unter Verwendung einer `for` Schleife

b. Entwickeln Sie das Programm unter Verwendung einer `do` Schleife und einer `break` Anweisung.

3.14 (*) Zusatzaufgabe: Pythagoräisches Dreieck *(einfacher Algorithmus, for-Schleife)*

Ein pythagoräisches Dreieck ist ein Dreieck, bei dem die 2 Seiten a, b und die Hypotenuse c **jeweils eine ganze Zahl** sind. Auch für solche Dreiecke gilt der Satz des Pythagoras $c^2 = a^2 + b^2$, wobei a, b Seiten und c die Hypotenuse. Schreiben Sie ein Programm mit drei ineinander geschachtelten `for`-Schleifen, das alle pythagoräischen Dreiecke findet, wobei `a`, `b` und `c` nicht größer als 500 sein dürfen. Also beispielsweise ist a=3, b=4 und c=5 ein pythagoräisches Dreieck. Lassen Sie sich alle gefunden Dreiecke mit den jeweiligen Seiten und Hyptonusen ausgeben.

Beispielsweise gibt das Programm folgende Dreiecke aus:

```
a: 3 b: 4  c: 5
a: 4 b: 3  c: 5
a: 5 b: 12  c: 13
…
```

3.15 (*) Rechenaufgaben *(Math random, Kontrollstrukturen)*

Entwerfen und schreiben Sie dazu ein Programm `Rechentest` für die Grundrechenarten, das die Rechenfähigkeiten eines Grundschülers testet. Das Programm erzeugt mit Hilfe eines Zufallsgenerators eine beliebige Anzahl von einfachen Rechenaufgaben mit Integer-Typen.

- Erzeugen Sie zufällig Zahlen im Wertebereich von 0 bis 32. Dazu benutzen Sie die Zufallsmethode: `Math.random()`, welche eine Zufallszahl (gleichverteilt) zwischen 0 und 1 liefert. Um Sie zu einer größeren Zufallszahl zu machen, multipliziert man diese Zahl mit dem gewünschten Zahlbereich (maximal gewünschter Wert) und führt eine explizite Typumwandlung um. So liefert zum Beispiel

  ```
  int x = (int) (Math.random()*200);
  ```

  eine natürliche Zahl zwischen 0 und 199.

- Ebenfalls zufällig wählen sie einen Operator. Auch diesen kann man mit dem Zufallsgenerator auswählen. Mit Hilfe der `switch`-Anweisung wird dann die Operation (+ , - , * , :) in Abhängigkeit von der Zufallsgröße gewählt.

- Beachten Sie bitte, dass nur korrekte Rechenaufgaben erzeugen (also keine 0 Division!!)

- Ihr Programm endet nie. D.h. es werden immer wieder neue Aufgaben erzeugt.

- Schließlich vergleichen Sie das Ergebnis mit der Berechnung des Computers und geben eine entsprechende Rückmeldung.

  Beispiel:

```
13-7=
>
6
Wow. Das war richtig

7+25=
>
22
Denke nochmals nach!!

28+20=
>
48
Wow. Das war richtig

28*29=
>
812
Wow. Das war richtig

52/26=
>
2
Wow. Das war richtig
```