for-Schleife

1. Aufgabe:

1 Tippe die Zeilen in BlueJ ein. Was macht das Programm?

```
public class C14ForSchleife {
    int i;
    for(i=0;i<=10;i=i+1 ) {
      System.out.println(n);
    }
  }
}
```

2 Schreibe kleine Programme:

- Von –5 bis 5 in Einerschritten, in einer Reihe aufgelistet, durch Kommas getrennt. Am Ende der Reihe soll kein Komma stehen.
- Von 10 bis 30 in Zweierschritten,
- Rückwärtszählen von 10 auf Null,
- Rückwärtszählen von 50 bis 0 in Viererschritten.

Nutze zur Angabe der Schrittweite die möglichen ***→Kurzschreibweise***.

Anleitung Anleitung Anleitung Anleitung Anleitung Anleitung Anleitung Anleitung Anleitung

for-Schleife

Mit Schleifen bzw. Wiederholungsanweisungen können Anweisungen mehrmals ausgeführt werden.
Die Schleifenvariable i kann vor der Schleife oder im Schleifenkopf deklariert werden.

```
int i;
for(i=0;i<=10;i=i+2 ){
  . . .;//Anweisungen
{
```

oder

```
for(int i=0;i<=10;i=i+2 ){
  . . .;//Anweisungen
{
```

`i = 0` gibt den Startwert der Schleife an.
`i <= 10` nennt die Bedingung, wann die Schleife abbrechen soll.
`i = i+2` legt die Schrittweite fest.

Schrittweite

```
i = i + 1
```

```
4    int i;
5    i=4;
6    System.out.println("Wert vorher :"+i);
7    i=i+1;
8    System.out.println("Wert nachher:"+i);
```

Der Wert der Variablen i wird um 1 erhöht und dann wieder der Variablen zugewiesen, damit wird der alte Wert überschrieben.

Kurzschreibweise

Statt `i=i+1` kannst du kürzer `i+=1` schreiben, ganz kurz `i++` .
Sinngemäß gilt: `i=i-1` oder `i-=1` oder `i--` .

2. Aufgabe:

1 Schreibe ein Programm, das die Unicode-Zeichen mit den Dezimalwerten 0 bis 255 ausgibt. Die Zeichen werden aneinander gereiht, getrennt durch Komma und Leerzeichen. Die Ausgabe muss mit der printf-Methode formatiert werden. **Lies dazu die Info auf http://www.hpg-speyer.de/index.php/unterrichtsfaecher/informatik/356** Klassenname: C14Unicode.

2 Schreibe ein Schleifenprogramm, bei dem der Anwender den Startwert, die Bedingung und die Schrittweite eingeben kann. Klassenname: C14ForEingabe.

Anleitung Anleitung Anleitung Anleitung Anleitung Anleitung Anleitung Anleitung Anleitung

Variablen im Schleifenkopf

```
int i , von;
for(i=von;i<=10;i=i+1 ){
```

Die festen Werte im Schleifenkopf können durch Variable ersetzt werden.

while-Schleife

Aufgabe:
- Ersetze in allen Programmen mit einer `for`-Schleife diese durch eine `while`-Schleife. Speicher die Programme unter C15While

Anleitung Anleitung Anleitung Anleitung Anleitung Anleitung Anleitung Anleitung Anleitung

while-Anweisung

```
5   int i=3;
6   while (i<10){
7     . . .;//Anweisungen
8    i=i+1;
9   }
```

5 Der Schleifenvariable i wird ein Startwert zugewiesen.
6 Nach dem Schlüsselwort `while` folgt in Klammern die Abbruchbedingung der Schleife.
8 Der Wert der Schleifenvariablen wird erhöht.

Schachtelung

Aufgabe:
- Gestalte Programme mit geschachtelten Schleifen so, dass die unten gezeigten Muster ausgegeben werden. Kreiere weitere eigene Muster.
Speicher die Programme unter dem Klassennamen C16Schachteln.

Konsole

```
0  1  2  3  4  5  6  7  8  9
0  1  2  3  4  5  6  7  8  9
0  1  2  3  4  5  6  7  8  9
0  1  2  3  4  5  6  7  8  9
0  1  2  3  4  5  6  7  8  9
```

Konsole

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

Konsole

```
1  2  3  4  5  6  7  8  9
2  3  4  5  6  7  8  9
3  4  5  6  7  8  9
4  5  6  7  8  9
5  6  7  8  9
6  7  8  9
7  8  9
8  9
9
```

Konsole

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

Konsole

```
1  2  3  4  5  6  7  8  9
2  3  4  5  6  7  8  9 10
3  4  5  6  7  8  9 10 11
4  5  6  7  8  9 10 11 12
5  6  7  8  9 10 11 12 13
6  7  8  9 10 11 12 13 14
7  8  9 10 11 12 13 14 15
8  9 10 11 12 13 14 15 16
9 10 11 12 13 14 15 16 17
```

Konsole

```
 0   1   2   3   4   5   6   7   8   9
10  11  12  13  14  15  16  17  18  19
20  21  22  23  24  25  26  27  28  29
30  31  32  33  34  35  36  37  38  39
40  41  42  43  44  45  46  47  48  49
50  51  52  53  54  55  56  57  58  59
60  61  62  63  64  65  66  67  68  69
70  71  72  73  74  75  76  77  78  79
80  81  82  83  84  85  86  87  88  89
90  91  92  93  94  95  96  97  98  90
```

Geschachtelte Schleifen

```
public class CSchachteln {
    int n,m;
    for(n=1;n<=9;n++) {
      //mögliche Anweisungen
      for(m=k;m<=l;m++) {
        //Gib den Wert von m aus
      }
      // mögliche Anweisungen
    }
}
```

Befindet sich im Anweisungsblock einer Schleife wiederum eine Schleife, spricht man von Schachtelung.

In der inneren Schleife muss mindestens eine Anweisung stehen.

In der äußeren Schleife können Anweisungen eingerichtet werden.

Verzweigung 1

Aufgabe: Tippe den Java-Code in BlueJ.
Experimentier mit dem Programm. Ändere dazu die Werte der Variablen m, n,

```
public class C17If {
    /*Vergleich von Zahlen*/
    int m = 20;
    int n = 30;
    boolean aussage = false;

    System.out.println("if-Abfrage");
    if (m<n){
      //Die Bedingung ist wahr
      System.out.println(m+" ist kleiner als "+n);
    }

//*****************************************************
    System.out.println();
    System.out.println("if-else-Abfrage");
    if (m>n){
      //Die Bedingung ist wahr
      System.out.println(m+" ist groesser als "+n);
    }
    else {
      //Die Bedingung ist falsch
      System.out.println(m+" ist kleiner als "+n);
    }

//*****************************************************
    /*Vergleich von boolean Werten*/
    System.out.println();
    System.out.println("if-else-Abfrage, boolean");
    if (aussage==true){
      //Die Bedingung ist wahr
      System.out.println("Heute scheint die Sonne.");
    }
    else {
      //Die Bedingung ist falsch
      System.out.println("Heute ist es bewoelkt.");
    }

}
```

if-Befehl

Mit dem **if**-Befehl wird abgefragt, ob eine Bedingung, sie steht in den runden Klammern (), erfüllt ist.
Ist die Bedingung wahr (true), wird der Anweisungsblock ausgeführt.

```
int m = 20;
int n = 30;

if (m<n){
  System.out.println(m+" ist kleiner als "+n);
}
```

Schema

```
if (Bedingung) {
  ...;
     //Anweisungen
  ...;
}
```

Der Anweisungsblock muss von geschwungenen Klammern { } umspannt werden!
Folgt nach der `if`-Zeile nur eine Anweisung, müssen keine Blockklammern {...} gesetzt werden.

Verzweigung 2

if-else Anweisung

Hier wird wieder eine Bedingung abgefragt. Ist sie erfüllt, dann wird das Eine getan, ansonsten das Andere.
Ist die Bedingung wahr (true), wird der `if`-Block ausgeführt, andernfalls der `else`-Block.

```
if (m>n){
  //Die Bedingung ist wahr
  System.out.println(m+" ist groesser als "+n);
}
else {
  //Die Bedingung ist falsch
  System.out.println(m+" ist kleiner als "+n);
}
```

Schema

```
if (Bedingung) {
   // Anweisungen
}
else {
   // Anweisungen
}
```

Schachtelung von Schleifen und Verzweigungen

Schleifen und Verzweigungen können beliebig miteinander kombiniert werden. Ein Beispiel zeigt, wie eine `if-else`-Anweisung in eine `while`-Schleife und dies wiederum in eine `for`-Schleife eingebettet ist.

```
for (i=1;i<=versuche;i++){
  . . . ; //Anweisung
  while(true){
    . . . ;//Anweisungen
    if (zahl > 5){
      . . . //Anweisungen
    }
    else {
       . . . //Anweisungen
    }
    . . . //Anweisungen
  }
}
```

Vergleichsoperatoren

Die Opratoren lefern wahr (true) oder falsch (false) zurück.

Operator	Bedeutung	gilt für
==	Gleichheit	Zahlen und Zeichenketten
!=	Ungleichheit	
>	größer	Zahlen
<	kleiner	
>=	größer oder gleich	
<=	kleiner oder gleich	

Logische Verküpfungs-Operatoren

Die opratoren lefern wahr (true) oder falsch (false) zurück.

Operator	Bedeutung	gilt für Verknüpfungen von Aussagen
&&	logisches UND	`if ( a > 2 && b < 10)` liefert wahr (true), wenn beide Aussagen wahr sind.
I I	logisches ODER	`if ( a > 2 && b < 10)` liefert wahr (true), wenn eine der Aussagen wahr ist.
!	logisches NICHT	Negiert den logischen Wert einer Aussage.