## Java07 − for-Schleife

Die grundlegenden Kontrollstrukturen in nahezu allen Programmiersprachen sind:

- die **`for`**-Schleife (Wiederholung mit fester Anzahl) **[heute]**,
- die **`while`**-Schleife (bedingte Wiederholung) [demnächst] und
- die **`if`**-Anweisung (bedingte Anweisung) [demnächst].

**Schritt 1 − BlueJ-Projekt „Java_03_Konsolenausgabe“ öffnen**

Öffne Dein BlueJ-Projekt „**Java_03_Konsolenausgabe**“ der letzten Stunde.

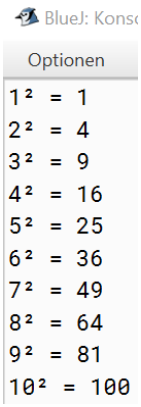
**Schritt 2 − Klasse „Schleife“ erstellen**

```
public class Schleife {
    public Schleife() {
    }
}
```

Erstelle eine neue Klasse „Schleife“ und öffne den Quelltext mit einem Doppelklick auf das Klassen-Symbol.

Lösche − bis auf den (leeren) Konstruktor „`public Schleife() {…}`“ − alles zwischen „`public class Schleife{`“ und der letzten geschweiften Klammer ganz unten. Das Ergebnis sollte so aussehen, wie Du es oben siehst.

**Schritt 3 − Methoden zur Erzeugung einer Liste von Quadratzahlen**




**1. AUFGABE: Methode „`public void forQuadratzahlen10()`“ erstellen**

Wir wollen auf der Konsole eine *Liste der ersten **10** Quadratzahlen* ausgeben.

Kopiere dazu den folgenden Code in den Quelltext unterhalb des Konstruktors, aber noch vor der letzten schließenden geschweiften „Klassen-Klammer“:

```
public void forQuadratzahlen10() {
    for (int i = 1; i <= 10; i++) {
        System.out.println(i + "² ist " + (i*i) + ".");
    }
}
```

Die hochgestellte 2, also „ ² “, erreichst Du mit der Tastenkombination „AltGr + 2“.

In der **`for`**-Schleife wird mit der Variablen **`i`** gezählt, wie oft der Schleifenrumpf ausgeführt werden soll.
Die Sichtbarkeit der Variablen **`i`** beschränkt sich auf den Rumpf der **`for`**-Schleife.
Außerhalb der **`for`**-Schleife kann auf diese Zählvariable nicht zugegriffen werden.
Aus diesem Grund heißt eine solche Variable **lokale Variable**.

Ablauf der for-Schleife:

1. Zuerst wird die Anweisung `int i = 1` ausgeführt.
   Die lokale Variable **`i`** wird als ganze Zahl („`int`“) deklariert und mit dem Wert 1 initialisiert.
2. Nun wird geprüft, ob die Bedingung `i <= 10` wahr ist.
3. Wenn sie wahr ist, wird der Rumpf der **`for`**-Schleife ausgeführt. Andernfalls wird das Programm mit den nach der **`for`**-Schleife ggf. stehenden Anweisungen fortgesetzt.
4. Nun wird die Anweisung `i++` ausgeführt.
   Der Wert der lokalen Variablen **`i`** wird dadurch um 1 erhöht (inkrementiert).
   Am Ende des Durchgangs wird die **`for`**-Schleife an der Stelle 2. wieder fortgeführt.

Bei der **`for`**-Schleife (Wiederholung mit fester Anzahl) muss vorher bekannt sein, wie oft die Schleife durchlaufen werden soll, d. h. hier, wie oft `System.out.println(i + "² ist " + (i*i) + ".");` ausgeführt werden soll. Die lokale Zählvariable `i` wird durch `i++` nach jedem Durchlauf um 1 erhöht.

WICHTIG: Erzeuge ein Objekt der Klasse „Schleife“ und **teste**, ob die Methode „tut, was sie soll“!

**2. AUFGABE: Methode „`public void forQuadratzahlenN(int n)`“ erstellen**

Wir wollen auf der Konsole eine *Liste der ersten **n** Quadratzahlen* ausgeben.

Beim Aufruf dieser Methode soll eine Abfrage erfolgen, wie viele Quadratzahlen erzeugt werden sollen.

Dies erreicht man, indem man im Methoden-Kopf in den **runden Parameter-Klammern** einen Parameter n vom Typ „ganze Zahl“ („`int`“) aufführt.

Wenn die Methode ausgeführt wird, kann der beim Methoden-Aufruf für diesen Parameter n eingegebene Wert verwendet werden, d. h. der Parameter kann im Code des Methoden-Rumpfes verwendet werden.

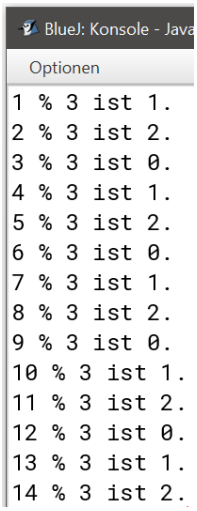
a) Kopiere den Code von „`public void forQuadratzahlen10()`“ (siehe oben) in den Quelltext unterhalb der Methode „`public void forQuadratzahlen10()`“, aber noch vor der letzten schließenden geschweiften „Klassen-Klammer“.

b) Ändere den Methoden-Kopf zu „`public void forQuadratzahlenN(int n)`“.

c) Überlege, an welcher Stelle im Methoden-Rumpf eine Änderung vorgenommen werden muss, damit die *ersten **n** Quadratzahlen* ausgegeben werden.

WICHTIG: Erzeuge ein Objekt der Klasse „Schleife“ und **teste**, ob auch diese Methode „tut, was sie soll“!

**Schritt 4 − Methode zur Erforschung des Operators „%“**




**3. AUFGABE: Methode „`public void xErforschen(…)`“ erstellen**

Mit dieser Methode wollen wir „automatisiert“ herausfinden, welche Wirkung das Rechenzeichen „%“ hat.

Beim Aufruf der Methode sollen zwei Parameter abgefragt werden:

1. die obere Grenze für die erste Zahl, d. h. die Zahl vor „%“ (in der Abb. rechts: 14)
2. die feste zweite Zahl, d. h. die Zahl nach „%“ (in der Abb. rechts: 3)

Der gesamte Methoden-Kopf ist also zum Beispiel:

```
public void xErforschen(int grenzeZahl1, int festeZahl2)
```

Die Namen für die Parameter können beliebig gewählt werden, möglichen wären auch „horst“ und „rosi“. Sinnvoll ist aber, Namen zu wählen, aus denen sich die vorgesehene Bedeutung des jeweiligen Parameters erschließt. Das erleichtert das Programmieren des Methoden-Rumpfes.

Der Datentyp der Parameter ist jeweils eine „ganze Zahl“ („`int`“).

Bei Parametern (in den runden Klammern), aber auch bei der Deklaration von Attributen oder von lokalen Variablen muss immer zuerst der Datentyp angegeben werden und danach der Name.

Füge nun diesen Methoden-Kopf unterhalb der Methode „`public void forQuadratzahlenN()`“ ein, aber noch vor der letzten schließenden geschweiften „Klassen-Klammer“.

Programmiere dann den Methoden-Rumpf, sodass beim Aufruf von „`xErforschen(…)`“ eine Liste wie oben auf der Konsole ausgegeben wird.

Tipps: i) Es wird eine for-Schleife benötigt.

ii) Es muss „irgendwo“ die Berechnung „`? % festeZahl2`“ erfolgen. (Was steht für „`?`“?)

WICHTIG: Erzeuge ein Objekt der Klasse „Schleife“ und **teste**, ob auch diese Methode „tut, was sie soll“!

Was bewirkt also das Rechenzeichen „%“?