**Unterrichtsstunde Schleifen**

Übersicht:

In der Stunde wird das Konzept von Schleifen erklärt und die Unterschiede zwischen den 3 Schleifentypen “while” “do-while” und “for” aufgezeigt. Dafür wird die Theorie für jede Schleife an dem gleichen Beispiel gelernt und anschließend weitere Aufgaben selbstständig von den Schülern gelöst.

Lernziel:

Nach dieser Stunde sollen die Schüler wissen, welche Typen von Schleifen es gibt, wie man sie einsetzt und in welchen Punkten sie sich unterscheiden. Des Weiteren sollen die Schüler ein Gefühl dafür entwickelt haben, welche Schleife wann am besten geeignet ist.

Zeitplan:

1. If-Bedingungen kurz wiederholen. 5min
2. Ableiten der while-Schleife von einer if-Bedingung. Unterschiede

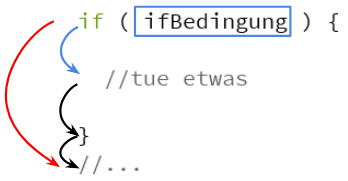
anhand eines Ablaufdiagramms aufzeigen. 10min

1. Anwenden anhand von Übung 1. 10min
2. Erklärung do-while-Schleife mit Ablaufdiagramm. 10min
3. Anwenden anhand von Übung 1. 10min
4. Praxisphase der Schüler Übung 2. 20min
5. For-Schleife erklärt anhand von Übung 1. 10min
6. Übung 3 15min
7. optional Übung 4 & 5.

Theorie:

Ablaufdiagramme können an der Tafel (alt. Beamer) gezeigt werden.

1. **Wiederholung von If-Bedingungen:**

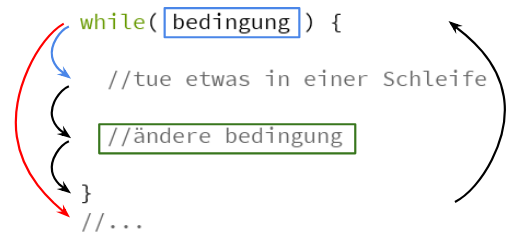


Ist die Bedingung erfüllt (**blau** ), wird der Code innerhalb der if-Bedingung ausgeführt.

Ist die Bedingung nicht erfüllt ( **rot** ), wird der Code innerhalb der if-Bedingung übersprungen.

1. **Ableiten der while Schleife von der If-Bedingung:**

Von der Syntax unterscheidet sich die while Schleife nur darin, dass das “if” durch ein “while” ersetzt wird.



Wie bei der If-Bedingung wird der Code in der while-Schleife ausgeführt, wenn die Bedingung erfüllt ist ( **blau** ), und wenn nicht ( **rot** ) nicht.

Jedoch wird nach der while-Schleife zurück zum Anfang der Schleife gesprungen und die Bedingung erneut geprüft, so dass der Code erneut ausgeführt wird.

Die Schleife ist beendet, wenn die Bedingung nicht mehr erfüllt ist ( **rot** ). In diesem Fall wird mit dem Code nach der Schleife fortgefahren.

Um nicht in einer Endlosschleife stecken zu bleiben, muss sichergestellt werden, dass die Bedingung irgendwann nicht mehr erfüllt ist, und das Programm normal fortfahren kann. Dafür muss die Bedingung innerhalb der Schleife angepasst werden ( **ändere bedingung** ), z.B. durch heraufzählen eines Integers.

1. **Übung 1 mit while Schleife:**

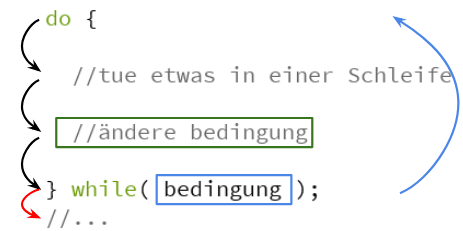
In der IDE am Beamer vorführen.

Wichtig:

* Schüler mit einbinden und fragen was geschrieben werden muss.
* aufzeigen/erklären, in welcher Reihenfolge der Code ausgeführt wird.

1. **Erklärung do-while-Schleife mit Ablaufdiagramm:**

Bei einer do-while-Schleife wird der Code zuerst ausgeführt und danach erst geprüft, ob die Schleife ein weiteres mal ausgeführt wird (erst “do” dann “while”).



Somit wird bei einer do-while-Schleife der Code immer mindestens einmal ausgeführt, auch wenn die Bedingung von Anfang an nicht erfüllt ist.

Auch hier muss die Bedingung angepasst werden ( **ändere bedingung** ), um sicher zu stellen, dass die Schleife irgendwann abgebrochen wird.

1. **Übung 1 mit do-while Schleife:**

In der IDE am Beamer vorführen.

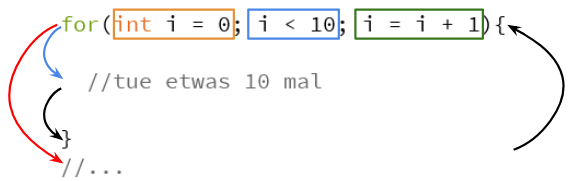
Wichtig:

* Schüler mit einbinden und fragen was geschrieben werden muss.
* aufzeigen/erklären, in welcher Reihenfolge der Code ausgeführt wird und dass die Schleife mindestens einmal ausgeführt wird.
  + kann dadurch erzielt werden, indem der erste Punkt bei

x = 0px gezeichnet wird, die Bedingung jedoch sagt, dass x größer als 0 sein muss.

* Gegenbeispiel: was passiert bei der While Schleife.

1. **For-Schleife:**



Die for-Schleife fasst alle wichtigen Teile für eine Schleife in den Klammern zusammen:

**Orange:** deklaration eines Integers für die Bedingung. Dieser wird meisten als “i” bezeichnet.

**Blau:** Bedingung

**Grün:** Anpassen der Bedingung, hier: Heraufzählen.

Ist die Bedingung erfüllt ( **Blau** ) wird der Code in der Schleife ausgeführt. Wenn nicht ( **rot** ), dann wird mit dem Code nach der Schleife fortgefahren. (wie bei der while-Schleife).

Das Anpassen der Bedingung wird von der for-Schleife am Ende dieser ausgeführt, wie als stände es in der letzten Zeile innerhalb der Schleife.

Danach wird erneut geprüft und die Schleife geht von vorne los.

1. **Übung 1 mit for-Schleife:**

In der IDE am Beamer vorführen.

Wichtig:

* Schüler mit einbinden und fragen was geschrieben werden muss.
* aufzeigen/erklären, in welcher Reihenfolge der Code ausgeführt wird.
* aufzeigen/erklären, dass die Bedingung erst am Ende der Schleife angepasst (hier hochgezählt) wird.

Übungen:

**1. Übung - Zeichnen von mehreren Punkten mit Schleifen**

Zeichne Punkte waagerecht in einem Abstand von 100px bis zu einer Grenze von X >= 500.

Dieses Beispiel wird zusammen mit allen Schleifentypen gelöst.

**2. Übung - while- und do-while-Schleifen**

a. Würfel, mit HIlfe einer while-Schleife, einen Würfel so lange, bis dieser eine 6 würfelt. Zähle die Anzahl an Versuchen, und gib diese aus.

Mit “Math.random()” erhält man einen zufälligen double-Wert zwischen 0 und 1.

b. Baue Übung 2a so um, dass eine do-while-Schleife verwendet wird. Welche Schleife eignet sich mehr und warum?

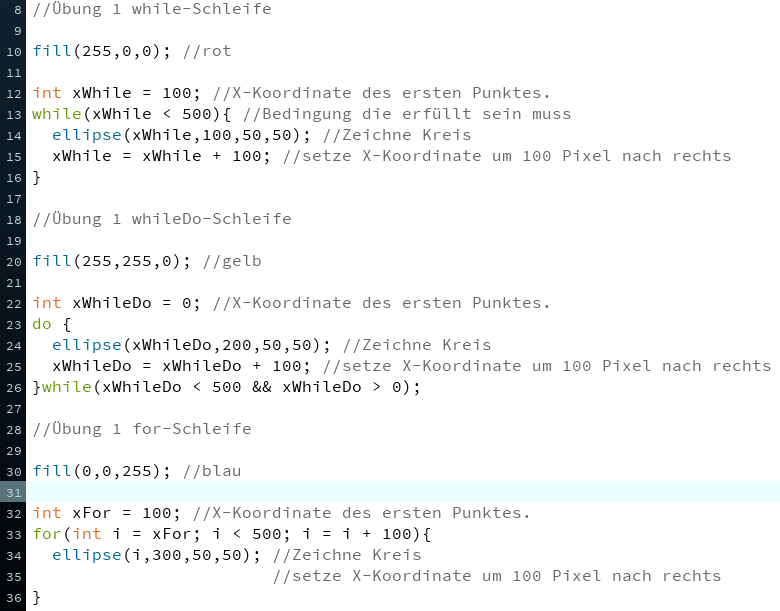
c. Baue 2a/2b so um, dass die Abbruchbedingung zwei hintereinander gewürfelte 6 sind. Zähle wieder die Anzahl der Versuche. (Einfacher: Abbruchbedingung bei der zweiten gewürfelten 6).

**3. Übung - for-Schleife**

Gebe für jeden int x zwischen -10 und 10 den y-Wert einer vorher definierten Funktion aus.

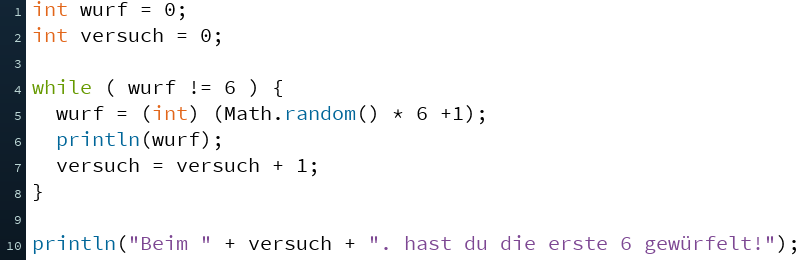
Lösungen:

**Übung 1:**

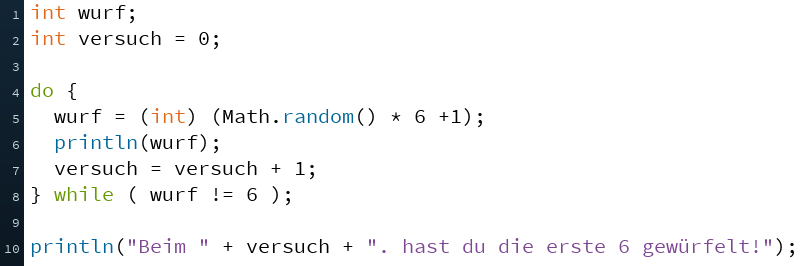


**Übung 2:**

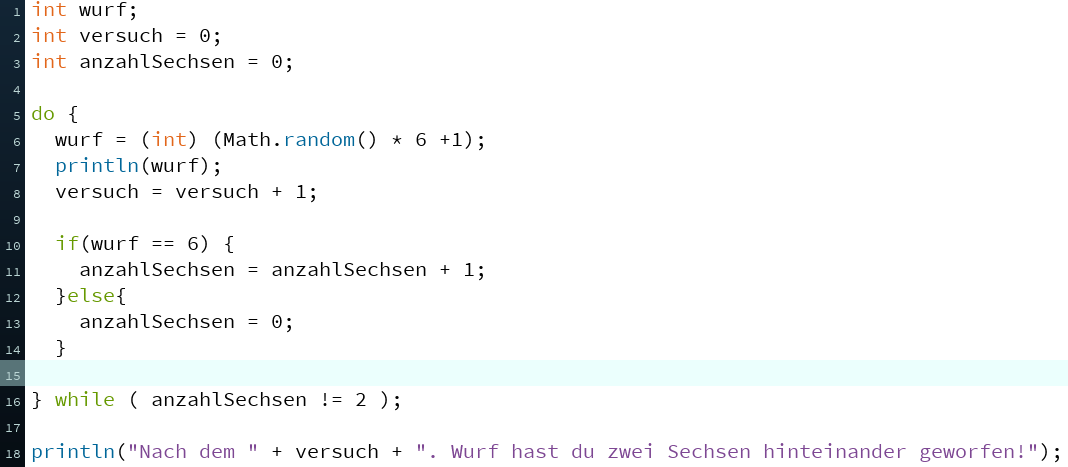
a.



b.



c.



**Übung 3:**

