3. Die Wiederholung (Iteration)/Schleifen

Im Verlauf eines Programms müssen manchmal eine oder mehrere Anweisungen wiederholt ausgeführt werden. Solche Wiederholungen nennt man auch (Programm-) Schleifen. In Java gibt es drei Arten von Schleifen:

- Die Anzahl der Schleifendurchläufe ist nicht von vornherein bekannt und wird während des Programmauslaufs durch eine **Bedingung** gesteuert, die bei jedem Schleifendurchlauf **am Schleifenanfang abgefragt** wird (**while-Schleife**).
- Die Anzahl der Schleifendurchläufe ist nicht von vornherein bekannt und wird während des Programmlaufs durch eine **Bedingung** gesteuert, die am **Schleifenende abgefragt** wird (**do-while-Schleife**).
- Die **Anzahl der Schleifendurchläufe** ist beim Erstellen des Programms von vornherein **bekannt (for-Schleife)**.

a) Die while-Schleife:

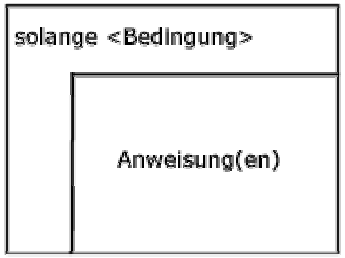
Bei der while-Schleife wird am Schleifenanfang eine Bedingung auf ihren Wahrheitsgehalt hin überprüft.

SOLANGE *Bedingung* FÜHRE AUS
Anweisung 1
Anweisung 2
....
Anweisung n
} Schleifenrumpf

Wenn die Ausführungsbedingung WAHR (true) ist, wird der Schleifenrumpf betreten und die Anweisung(en) wird (werden) ausgeführt. Ist die Ausführungsbedingung FALSCH (false), so wird der Schleifenrumpf nicht betreten und das Programm mit der dahinter folgender, Anweisung fortgesetzt. Es kann demnach sein, daß die while-Schleife kein einziges Mal durchlaufen wird. Man nennt daher die while-Schleife eine **abweisende Schleife**.

Wichtig: Man muss unbedingt darauf achten, daß die Schleife auch tatsächlich nach einer endlichen Anzahl von Durchläufen verlassen wird und keine Endlosschleife entsteht. Dies bedeutet, daß der Wahrheitswert der Ausführungsbedingung nach endlich vielen Durchläufen den Wert FALSCH annehmen muss. Erreicht wird das durch eine entsprechende Anweisung im Schleifenrumpf.

Struktogramm:



Formulierung in JAVA: **while (**Bedingung) Anweisung;

Ist die Anweisung aus mehreren Einzelanweisungen zusammengesetzt, so müssen. diese durch { und } zusammengefasst werden,

Beispiele:

```
public class WhileDemo{

  public static void main(String args[]){
    // Initialisierung
    int x=20;

    // Schleife
    while(x>1){
      System.out.println(x);
      x=x/2;
    }
  }
}
```

Das Programm gibt die Zahlen 20,10,5,2 in jeweils einer neuen Zeile aus.
Außerdem sind die Zeilen `// Initialisierung` und `// Schleife` neu. Diese Zeilen sind Kommentare. Alles was hinter // steht, wird beim Kompilieren ignoriert.

Wenn in dem Beispiel die Zeile x=x/2 fehlen würde, so würde die Schleife endlos wiederholt. Wäre die Initialisierung x=1 anstelle von x=20, so würde die Schleife kein einziges Mal durchlaufen.

Aufgabe: Ändere das Programm so, dass die Zahlen 30,27,24,21,18,15,12,9,6,3 ausgegeben werden.

```
public class WhileDemo2{

  public static void main(String args[]){
    int i=0;
    while(i<10){
      System.out.println(i);
      ++i;
    }
  }

}
```

Dieses Programm gibt die Zahlen 0, 1, 2, 3, 4, 5, 6, 7, 8, 9 in jeweils einer neuen Zeile aus.

Aufgaben:

1. Schreibe ein Java-Programm, das eine Wertetabelle für den Sinus ausgibt. Den Sinus einer Zahl x erhält man durch Math.sin(x). x muss man dabei im Bogemaß angeben.
2. Die Summe aller Zahlen bis zu einer eingegebenen Zahl soll bestimmt werden.
3. Die Summe aller ganzen Zahlen zwischen den ganzen Zahlen "AnfZahl" und "EndZahl" (beide sind einzugeben) einschließlich dieser beiden Werte soll berechnet werden.
4. Berechne analog zur vorigen Aufgabe das Produkt aller ganzen Zahlen von einem einzugebenden Anfangswert bis zu einem Endwert.
5. Der Computer soll nach Eingabe einer natürlichen Zahl kleiner als 20 (Kontrolle durch das Programm!) die nächsten 15 natürlichen Zahlen zusammen mit ihrem Quadrat ausgeben. Struktogramm und Java-Programm.
6. Es soll geprüft werden, ob welche Zahlen x in einem vorgegebenen Bereich die Ungleichung $|x - 7| < 3$ erfüllen .

