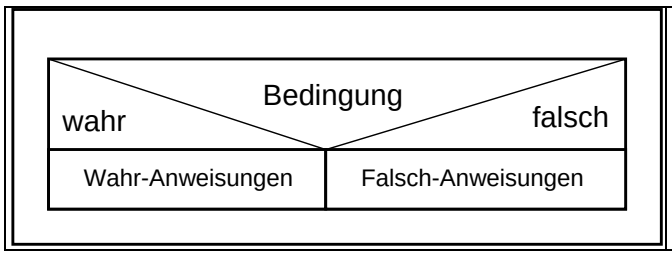
Gottlieb-Daimler-Schule 2
Technisches Schulzentrum Sindelfingen
mit Abteilung Akademie für Datenverarbeitung

Inhaltsverzeichnis

1 Verzweigung – die if-Anweisung



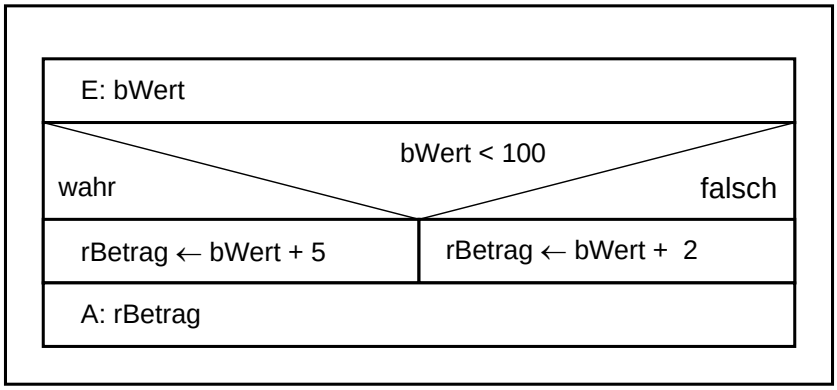

```
1  if (Bedingung)
2  {
3    WAHR-Anweisung(en);
4  }
5  else
6  {
7    FALSCH-Anweisung(en);
8  }
```

Syntax:
1 die **`if`**-Anweisung beginnt mit dem Schlüsselwort **`if`**,
die Bedingung wird in **runde Klammern** eingeschlossen, die Prüfung der Bedingung liefert ein Ergebnis vom Datentyp `boolean`
am Ende steht **kein Semikolon**

3 Hat die Prüfung das Ergebnis `true`, werden alle folgenden Anweisungen ausgeführt, die in geschweifte Klammern eingeschlossen sind (**2, 4**).

5 Hat die Prüfung das Ergebnis `false`, werden alle folgenden Anweisungen (**7**) nach dem Schlüsselwort `else` ausgeführt, die in geschweifte Klammern eingeschlossen sind (**6, 8**).

Beispiel:
Ein Versandhaus berechnet die Höhe des Portos und der Verpackung in Abhängigkeit vom Bestellwert: unter 100 € 5 €, 100 und mehr € 2 €:



Zugehöriges Java-Programm:

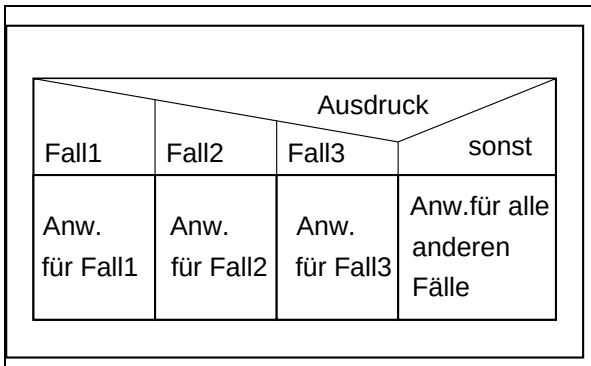
```
public class Bestellwert
{
  public static void main (String [] arg)
  {
      int iBestellwert;
      int iRechnungsbetrag = 0;

      iBestellwert = Eingabe.getInt("Bestellwert = ");

      if (iBestellwert < 100)
      {
        iRechnungsbetrag  = iBestellwert  + 5;
      }
      else
      {
        iRechnungsbetrag  = iBestellwert  + 2;
      }
      System.out.println("Rechnungsbetrag = "+iRechnungsbetrag);
  }
}
```

2 Mehrfachverzweigung – die switch-Anweisung

Mit einer `switch`-Anweisung wird keine Bedingung getestet, sondern der Wert einer Variablen oder eines Ausdrucks.




```
1 switch (Ausdruck)
2 {
3   case Wert_1: Anweisung(en) Wert 1;
4                break;
5   case Wert_2: Anweisung(en) Wert 2;
6                break;
7   case Wert_3: Anweisung(en) Wert 3;
8                break;
9   default    : sonst-Anweisung(en);
10               break;
11 }
```

Syntax:

1 Das Schlüsselwort `switch` leitet eine mehrseitige Auswahl ein. Der zu prüfende Ausdruck steht in runden Klammern. Zugelassen sind nur die Datentypen `char`, `byte`, `short` und `int`, sowie – seit Java 7 – `String`.

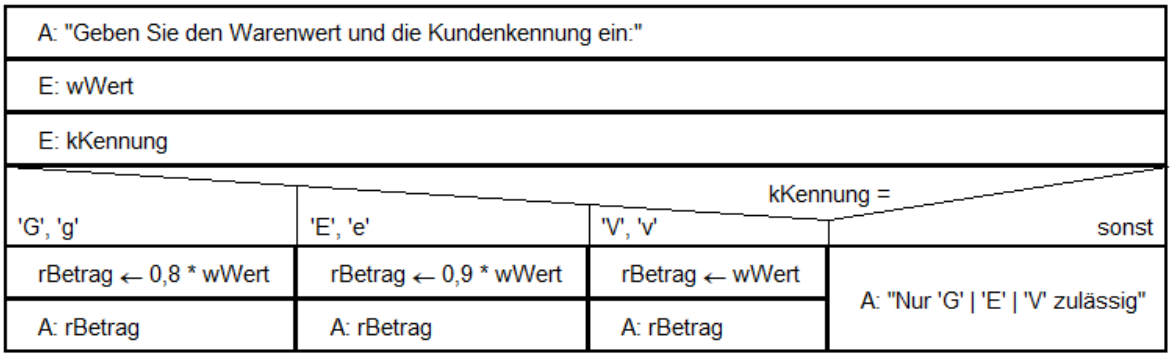
2 geschweifte Klammern begrenzen die `case`-Anweisungen (2, 11)

3 jeder Test des Ausdrucks beginnt mit dem Schlüsselwort `case`, dahinter darf nur ein einzelner Auswahlwert angegeben werden. Jeder Wert darf nur einmal geprüft werden. Nach dem Wert steht ein Doppelpunkt.

4 Jede `case`-Anweisung *kann* mit dem Schlüsselwort `break` beendet werden. Dadurch werden alle folgenden Anweisungen übersprungen

9 Stimmt der Wert des Ausdrucks mit keinem Auswahlwert überein, so werden Anweisung hinter der (optionalen) `default`-Anweisung ausgeführt.

Beispiel: In Abhängigkeit von der Kundenkennung werden verschiedene Rabatte gewährt: Großhändler (G) erhalten 20%, Einzelhändler (E) 10% und Endverbraucher (V) 0%.



Zugehöriges Java-Programm:

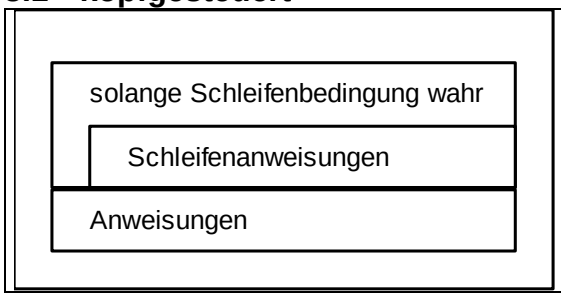
```
public class Rabatte
{
  public static void main (String [] arg)
  {
    int    iBestellwert;
    double dRechnungsbetrag;
    char   cKundenkennung;

    iBestellwert     = Eingabe.getInt("Bitte den Bestellwert eingeben:");
    dRechnungsbetrag = 0.0;
    cKundenkennung   = Eingabe.getChar("Bitte die Kundenkennung(G,E,V):");

    switch (cKundenkennung)
    {
      case 'G':
      case 'g':dRechnungsbetrag = iBestellwert * 0.8;
               System.out.printf("Rechnungsbetrag = %8.2f%n", dRechnungsbetrag);
               break;
      case 'E':
      case 'e':dRechnungsbetrag = iBestellwert * 0.9;
               System.out.printf("Rechnungsbetrag = %8.2f%n", dRechnungsbetrag);
               break;
      case 'V':
      case 'v':dRechnungsbetrag = iBestellwert;
               System.out.printf("Rechnungsbetrag = %8.2f%n", dRechnungsbetrag);
               break;
      default: System.out.println("Nur 'G' | 'E' | 'V' zulässig\n");
               break;
    }
  }
}
```

3 Wiederholung

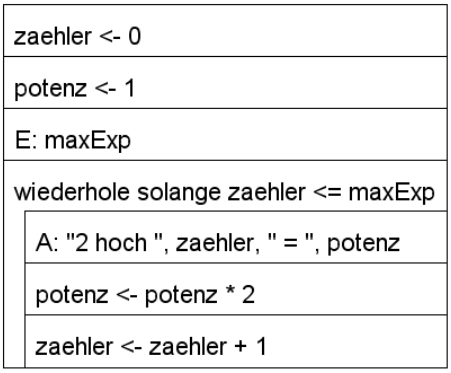
3.1 kopfgesteuert




```
1 while (Bedingung)
2 {
3   Schleifen-Anweisungen;
4 }
5 Weitere-Anweisungen;
```

Syntax:
1 Das Schlüsselwort `while` leitet die `while`-Schleife ein, die Schleifenbedingung wird in runde Klammern eingeschlossen
2 die Schleifenanweisungen werden in geschweifte Klammern eingeschlossen (**2, 4**)

Beispiel:
Zweier-Potenzen berechnen bis zu einem bestimmten maximalen Exponenten.



Zugehöriges Java-Programm:

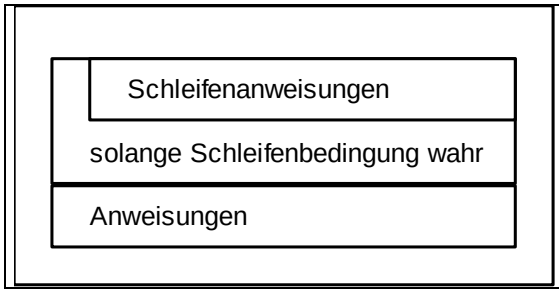
```
public class ZweierPotenzWhile
{

      public static void main(String[] args)
      {
            int zaehler = 0;
            int potenz = 1;
            int maxExp;
            System.out.println("Geben Sie den max. Exponenten ein:");
            maxExp =  Eingabe.getInt();

            while (zaehler <= maxExp)
            {
                  System.out.printf("2^%2d = %11d\n", zaehler, potenz);
                  zaehler = zaehler + 1;
                  potenz = potenz * 2;
            }
      }

}
```

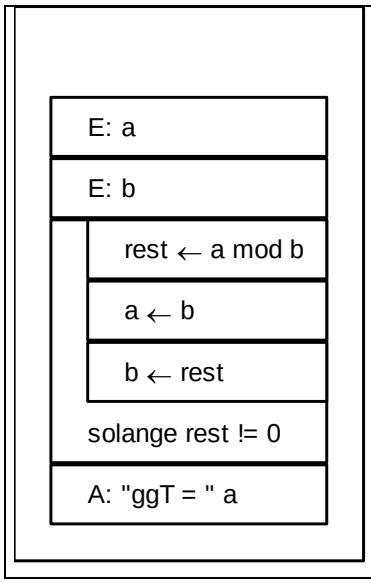
3.2 fußgesteuert




```
1 do
2 {
3   Schleifen-Anweisungen;
4 } while (Bedingung);
5 Anweisungen;
```

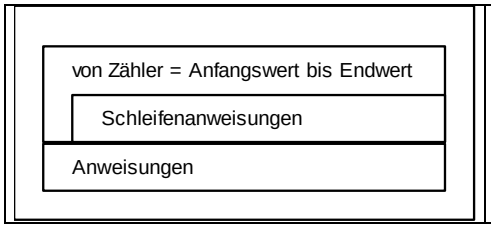
Syntax:
1 Das Schlüsselwort do leitet die `do-while`-Schleife ein
2 die Schleifenanweisungen werden in geschweifte Klammern eingeschlossen (**2, 4**)
4 die `while`-Anweisung wird mit einem Semikolon abgeschlossen

Beispiel:
Größter gemeinsamer Teiler (ggT) aus zwei natürlichen Zahlen berechnen.




```
public class GGTBerechnen
{
  public static void main (String [] arg)
  {
    int a;
    int b;
    int rest;
    a = Eingabe.getInt("Zahl 1 = ");
    b = Eingabe.getInt("Zahl 2 = ");
    do
    {
      rest = a % b;
      a = b;
      b = rest;
    } while (rest != 0);
    System.out.printf("ggT = %d", a);
  }
}
```

3.3 Zählschleife

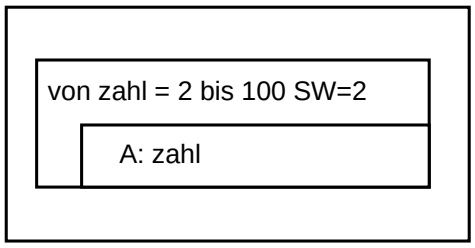



```
1 for (Initialisierung; Bedingung; Aktualisierung)
2 {
3   Schleifen-Anweisungen;
4 }
```

Syntax:

1 Das Schlüsselwort `for` leitet die `for`-Anweisung ein.
in runden Klammern eingeschlossen folgen 3 Teile, die durch Semikolon voneinander getrennt sind:
die *Initialisierung*, sie wird vor dem ersten Schleifendurchlauf ausgeführt.
die *Schleifenbedingung* wird bei jedem Durchlauf geprüft und liefert `true` oder `false`.
und die *Aktualisierung* der Schleifenvariablen. Sie wird am Ende jedes Schleifendurchlaufs ausgeführt, bevor die Bedingung geprüft wird.
Jeder dieser Teile kann auch fehlen, das Semikolon muss jedoch geschrieben werden.

Beispiel:
Alle geraden Zahlen von 2 bis 100 ausgeben:




```
public class ZahlenAusgeben
{
  public static void main (String [] arg)
  {
    int zahl;

    for (zahl = 2; zahl <= 100; zahl = zahl+2)
    {
      System.out.println("Zahl: "+zahl);
    }
  }
}
```

Programmierkonvention: Die for-Anweisung darf nur in solchen Fällen verwendet werden, wenn vor dem Schleifenbeginn die Anzahl der Schleifendurchläufe feststeht. Falls dies nicht gegeben ist, muss einer der anderen Schleifenarten verwendet werden. Insbesondere gilt, dass die Zählvariable im Schleifenkörper nicht verändert werden darf. Dies würde die Lesbarkeit des Programms verringern und Programmierfehler begünstigen.