Situationsbeschreibung Nettorechner

Frau Akurat aus dem Rechnungswesen meldet sich.

Liebe Kollegen,
um Eingabefehler zu vermeiden, benötige ich ein Programm, welches aus einem vorhandenen Bruttopreis den Nettopreis berechnet und auch einheitlich für meine anderen Programme einsetzbar ist.

Viele Grüße!

Herr Schäring (Abteilungsleiter Software) meldet sich daraufhin:

Hallo,
bitte setzen Sie das geforderte Programm gleich so um, dass wir es leicht in anderen Aufträgen *wiederverwenden* können.

Grüße!

Wir benötigen also eine **Methode*,** an welche wir einen Bruttopreis **übergeben** und welche uns den Nettopreis **zurückgibt**. Anders ausgedrückt: Eine Art Blackbox, in die ein Bruttopreis reingesteckt wird, dann arbeitet die Blackbox und spuckt am Ende den korrekten Nettopreis aus.

Berechnung des Nettobetrages (bei Umsatzsteuer von 19%)

Was passiert in der Blackbox?

Bruttopreis	≙ 119%	
Bruttopreis / 119	≙ 1%	
(Bruttopreis / 119) * 100	≙ 100%	≙ Nettopreis
Nettopreis	= Bruttopreis / 1,19	

* Die Begriffe "Methode, Funktion, Unterprogramm und Prozedur" bedeuten grundsätzlich dasselbe: ein aufrufbares Unterprogramm innerhalb eines größeren Programms. Eine Definition zu finden, die alle kleinen Unterschiede dieser Begriffe erfasst, ist schwer, da sie nicht durchgehend in allen Programmiersprachen verwendet werden. Den Begriff "Methode" verwendet man fast ausschließlich in objektorientierten Sprachen (wie C#) und bezieht sich auf eine Funktion, die einem Objekt zugeordnet ist - aber dazu später mehr. 😉

Informationstext zu Methoden

Mit Hilfe von Methoden können komplexe Programmabläufe in kleinere Teilprobleme zerlegt werden. Methoden liefern einen Wert an die aufrufende Stelle zurück (**Rückgabewert**) und können beliebig viele **Übergabeparameter** erhalten.

Um Methoden in C# zu implementieren, müssen drei Aspekte beachtet werden:

1. Der **Methodensaufruf** (z. B. aus der main-Funktion heraus).
2. Die **Methodendefinition** und der **Methodencode**, also die eigentliche Implementation der Methode.

Der **Methodenaufruf** erfolgt immer zusammen mit dem Methodennamen und den Übergabewerten – **in der main- bzw. der aufrufenden Methode**. Der Rückgabewert der Methode kann an der aufrufenden Stelle weiterverwendet werden (z. B. zuweisen an eine Variable). Die allgemeine Syntax eines Methodenaufrufs lautet:

```
                              Methodenname      ( Übergabewert, … );
                                   ⬇                    ⬇
Beispiel:     int iErgebnis =   potenzBerechnen   (        5        );
                  ⬆
```

beispielhafte Verarbeitung des Rückgabewertes, hier: Speichern des Rückgabewertes in eine Variable

Die **Methodendefinition** ist die Implementation der Logik der Methode - d. h. der Methodenrumpf wird festgelegt. Sie erfolgt **nach der main-Funktion**. Am Ende des Methodenrumpfs steht das Schlüsselwort **return** gefolgt von einem Rückgabewert. Dieser wird an die aufrufende Stelle zurückgeliefert.

Die allgemeine Syntax lautet:

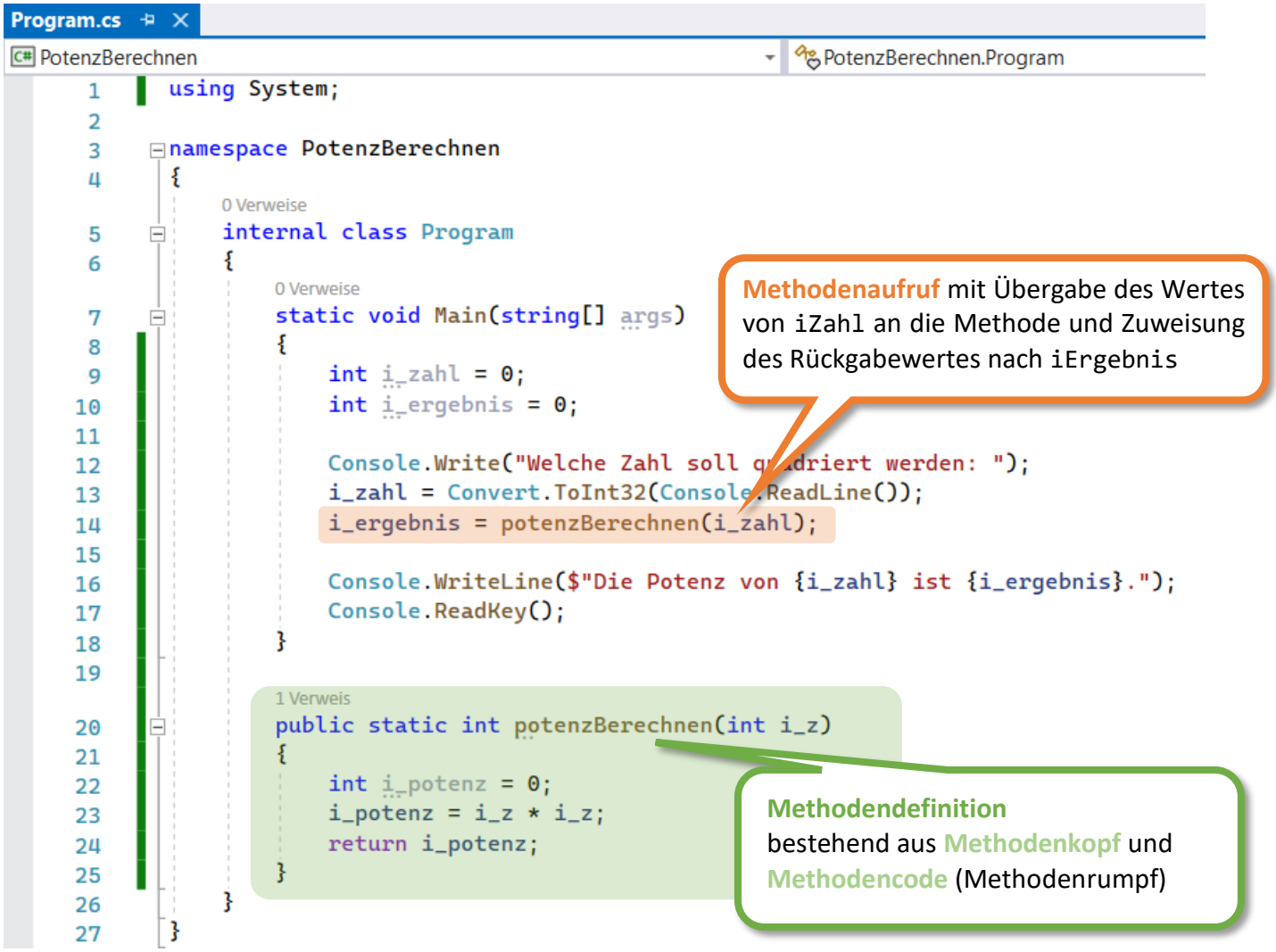
```
Modifizierer Rückgabedatentyp Methodenname (Übergabedatentyp Übergabename, … )
{
      Anweisungsblock;
      return Rückgabewert;
}
```

Methodenkopf

Methodenrumpf (der Code der Fkt.)

Beispiel zu Methoden in C#

(Programm zur Potenzberechnung mit Hilfe einer Methode)




```
1   using System;
2
3   namespace PotenzBerechnen
4   {
        0 Verweise
5       internal class Program
6       {
            0 Verweise
7           static void Main(string[] args)
8           {
9               int i_zahl = 0;
10              int i_ergebnis = 0;
11
12              Console.Write("Welche Zahl soll quadriert werden: ");
13              i_zahl = Convert.ToInt32(Console.ReadLine());
14              i_ergebnis = potenzBerechnen(i_zahl);
15
16              Console.WriteLine($"Die Potenz von {i_zahl} ist {i_ergebnis}.");
17              Console.ReadKey();
18          }
19
            1 Verweis
20          public static int potenzBerechnen(int i_z)
21          {
22              int i_potenz = 0;
23              i_potenz = i_z * i_z;
24              return i_potenz;
25          }
26      }
27  }
```

Achtung!

- Da die Übergabevariable (`i_z`) und der Methodenrumpf {…} einen „eigenen Block“ darstellen, stehen Variablen, die innerhalb dieses Blockes – also lokal – definiert werden, auch nur genau dort zur Verfügung (z. B. steht `i_potenz` nur innerhalb der Methoden zur Verfügung).

- Bei der Übergabe wird ***nur der Wert*** (also der Inhalt) der Variable `i_zahl` (siehe Mitte main-Funktion) an die Methode übergeben. In der Methode wiederum existiert eine eigenständige Variable namens `i_z` (Übergabevariable), welche nun denselben Wert wie `i_zahl`, aber eine eigene andere Speicheradresse hat. Dies bedeutet, der *Variableninhalt* wurde ***nur kopiert***. Innerhalb der Methode kann nicht auf `i_zahl`, außerhalb der Methode nicht auf `i_z` zugegriffen werden.

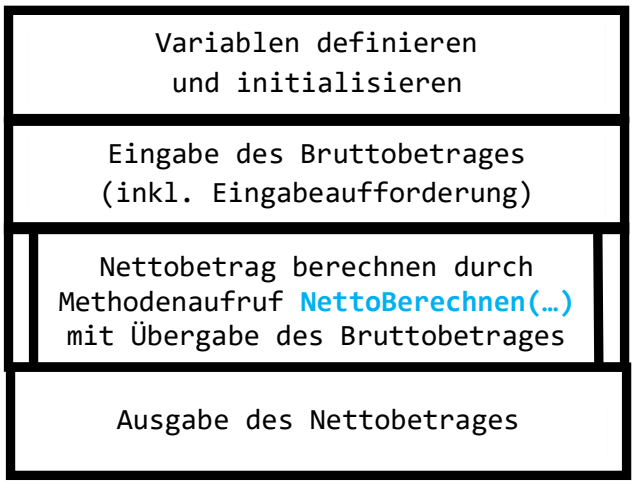
Um Missverständnisse zu vermeiden, sollten die Variablen innerhalb der Methode andere Namen (bspw. **`i_z`**) erhalten, als außerhalb der Methode (bspw. **`i_zahl`**).

Darstellung des Nettorechners im Struktogramm

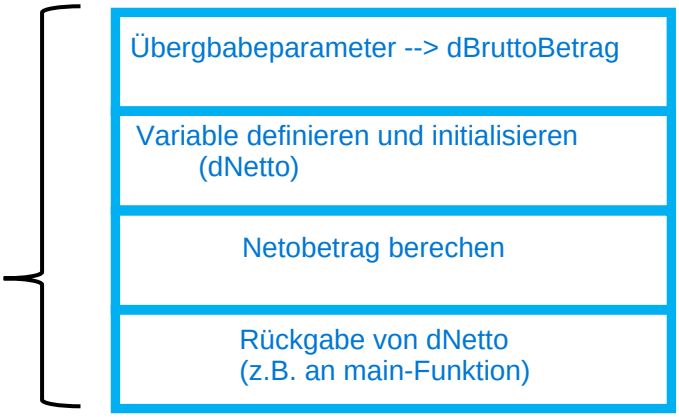
In einem Struktogramm werden Methodenaufrufe immer mit zwei vertikalen Strichen am Rand eines Anweisungsblockes dargestellt.

Am Beispiel des Nettorechners:

main-Einstiegspunkt:



***NettoBerechnen*-Methoden:**



Zusätzliches Beispiel zu Methoden:

Nehmen wir eine Methode zur Berechnung der Kreisfläche.

- **Methodenaufruf:** Die Methode bekommt als Argument (Übergabewert) den Kreisradius (bspw. 5cm) übergeben, rechnet die Kreisfläche aus und liefert sie gleichzeitig als Rückgabewert zurück. Der Rückgabewert (78,5) der Methode wird anschließend in der Variablen `d_ergebnis` abgelegt.

```
double d_ergebnis = kreisflaeche(5);    //kreisflaeche(5) wird zu → 78.5
```

Datentypen haben beim Methodenaufruf nichts verloren

- **Methodendefinition:** Der Methodenaufruf bewirkt, dass der Wert `5` (aus der Übergabe) in die Variable `d_radius` kopiert wird. Anschließend wird die Rechnung `3.14 * d_radius * d_radius` (also: `3.14 * 5 * 5`) durchgeführt. Das Ergebnis dieser Rechnung (`78.5`) wird dann vom `return`-Befehl an die aufrufende Methode zurückgegeben.

```
double kreisflaeche(double d_radius)
{
   return 3.14 * d_radius * d_radius;  //Rückgabe des Wertes aus 3.14 * 5 * 5
}
```

Übung zu Methoden – Ablauf von Methodenaufrufen:

1. Zeichnen Sie den Aufruf einer Methode mit Hilfe von Pfeilen und Nummerierungen in nachstehendes Schaubild ein.

Hauptprogramm

```
static void main()
{
      int a = 0;
      //Anweisung 1
      //Anweisung 2
      ...
      a = method();
      ...
      //Anweisung 3
}
```

Methode

```
static int method()
{
      //Anweisung A
      //Anweisung B
      ...
      return Ergebnis;
}
```

2. Auf der nächsten Seite ist ein Beispielprogramm gegeben. Es dient zur MwSt-Berechnung und setzt folgende Anforderungen mit Hilfe von vier Methoden um:

   - Bildschirmausgabe eines Menüs
   - Eingabe der Werte für die Variablen
   - Berechnung der Mehrwertsteuer
   - Ergebnisausgabe

Tragen Sie im beigefügten Programmschema (nächste Seite) den Programmablauf nach dem Start der Hauptfunktion mit Pfeilen ein und nummerieren sie diese wie im oben gezeigten Beispiel der Aufgabe 1.

Beispielprogramm zur MwSt-Berechnung (in Pseudocode)