Methoden

Methoden dienen in erster Linie dazu, Programmteile mehrmals benutzerbar zu machen.

Beispiel: Das untere Programm rechnet eine Körpergröße in Zentimeter auf die Einheit Fuß[1] um. Am Beginn wird der Benutzer begrüßt und am Ende verabschiedet. Sowohl am Beginn, als auch am Ende wird eine Copyright-Meldung mit dem aktuellen Datum und der aktuellen Uhrzeit ausgegeben.

Programmausgabe auf der Konsole:

```
HALLÖCHEN!
©SLOG, Wien 2021
Datum und Uhrzeit: 27.02.2021 21:04:42
Gib deine Körpergröße in Zentimeter ein: 163
Du bist 5,35 Fuß groß.
©SLOG, Wien 2021
Datum und Uhrzeit: 27.02.2021 21:05:11
AUF WIEDERSEHEN!
```

```
static void Main(string[] args)
{
    Console.OutputEncoding = Encoding.Unicode;

    // Willkommensmeldung
    Console.WriteLine("HALLÖCHEN!");
    Console.ForegroundColor = ConsoleColor.Yellow;
    Console.WriteLine("\u00A9SLOG, Wien 2021"); // U+00A9 ist das Copyrightzeichen
    Console.WriteLine("Datum und Uhrzeit: " + DateTime.Now);
    Console.ResetColor();

    // Eingabe und Berechnung
    Console.Write("Gib deine Körpergröße in Zentimeter ein: ");
    int cm = int.Parse(Console.ReadLine());
    double feet = Math.Round(cm * 3.28084e-2, 2); // gerundet auf zwei Stellen
    Console.WriteLine("Du bist " + feet + " Fuß groß.");

    // Verabschiedung
    Console.ForegroundColor = ConsoleColor.Yellow;
    Console.WriteLine("\u00A9SLOG, Wien 2021");
    Console.WriteLine("Datum und Uhrzeit: " + DateTime.Now);
    Console.ResetColor();
    Console.WriteLine("AUF WIEDERSEHEN!");

    Console.ReadKey();
}
```

Dieses Programm hat zwei identische Teile (siehe { }). In so einem Fall ist es sinnvoll und notwendig (!) eine Methode zu definieren, die die Programmzeilen mit den entsprechenden Zeilen enthält.

```
static void CopyrightMeldungAusgeben()
{
    Console.ForegroundColor = ConsoleColor.Yellow;
    Console.WriteLine("\u00A9SLOG, Wien 2021"); // U+00A9 ist das Copyrightzeichen
    Console.WriteLine("Datum und Uhrzeit: " + DateTime.Now);
    Console.ResetColor();
}
```

[1] https://de.wikipedia.org/wiki/Fu%C3%9F_(Einheit)

An den beiden Stellen wo bisher die Programmzeilen gestanden sind, schreibt man den Namen der Methode und `();` Eine solche Stelle nennt man *Methodenaufruf* oder *Aufruf der Methode* *`CopyrightMeldungAusgeben`*.

Das Main wird dadurch wesentlich kürzer, weil es keinen doppelten Programmcode mehr gibt und die Anweisungen für das Ausgeben der Copyright-Meldung sind zusammengefasst.

```
static void CopyrightMeldungAusgeben()
{
    Console.ForegroundColor = ConsoleColor.Yellow;
    Console.WriteLine("\u00A9SLOG, Wien 2021"); // U+00A9 ist das Copyrightzeichen
    Console.WriteLine("Datum und Uhrzeit: " + DateTime.Now);
    Console.ResetColor();
}

static void Main(string[] args)
{
    Console.OutputEncoding = Encoding.Unicode;

    // Willkommensmeldung
    Console.WriteLine("HALLÖCHEN!");
    CopyrightMeldungAusgeben();

    // Eingabe und Berechnung
    Console.Write("Gib deine Körpergröße in Zentimeter ein: ");
    int cm = int.Parse(Console.ReadLine());
    double feet = Math.Round(cm * 3.28084e-2, 2); // gerundet auf zwei Stellen
    Console.WriteLine("Du bist " + feet + " Fuß groß.");

    // Verabschiedung
    CopyrightMeldungAusgeben();
    Console.WriteLine("AUF WIEDERSEHEN!");

    Console.ReadKey();
}
```

Das Programm gibt auf der Konsole genau das gleiche aus wie zuvor.

Methoden mit einem Inputparameter

Die obere `CopyrightMeldungAusgeben` -Methode macht bei jedem Aufruf exakt das gleiche. Sehr häufig möchte man Methoden aber einen oder mehrere Werte mitgeben, damit sie mehr können.

Beispielsweise wäre es doch nett, wenn wir in unserem Beispiel den Text der Copyright-Meldung in verschiedenen Farben ausgegeben könnten. Dazu muss man einen Inputparameter definieren.

Parameter – es gibt auch noch andere als Inputparameter – stehen bei Methoden immer innerhalb der Klammer.

Der Datentyp für die Konsolenfarbe heißt `ConsoleColor`. Den Wert erhält ein Inputparameter immer während der Programmausführung

Datentyp und Name des Inputparameters

```
static void CopyrightMeldungAusgeben( ConsoleColor farbe )
{
    Console.ForegroundColor = farbe;
    Console.WriteLine("\u00A9SLOG, Wien 2021"); // U+00A9 ist das Copyrightzeichen
    Console.WriteLine("Datum und Uhrzeit: " + DateTime.Now);
    Console.ResetColor();
}


static void Main(string[] args)
{
    Console.OutputEncoding = Encoding.Unicode;

    // Willkommensmeldung
    Console.WriteLine("HALLÖCHEN!");
    CopyrightMeldungAusgeben(ConsoleColor.Yellow);

    // Eingabe und Be
    Console.Write("Gi
    int cm = int.Pars
    double feet = Mat
    Console.WriteLine

    // Verabschiedung
    CopyrightMeldungAusgeben(ConsoleColor.Red)
    Console.WriteLine("AUF WIEDERSEHEN!");

    Console.ReadKey()
}
```

Bei der Programmausführung!

= ConsoleColor.Yellow

```
static void CopyrightMeldungAusgeben( ConsoleColor farbe )
{
    Console.ForegroundColor = farbe;
    Console.WriteLine("\u00A9SLOG, Wien 2021"); // U+00A9 ist das Copyrigh
    Console.WriteLine("Datum und Uhrzeit: " + DateTime.Now);
    Console.ResetColor();
}
```

= ConsoleColor.Red

```
static void CopyrightMeldungAusgeben( ConsoleColor farbe )
{
    Console.ForegroundColor = farbe;
    Console.WriteLine("\u00A9SLOG, Wien 2021"); // U+00A9 ist das Copyrigh
    Console.WriteLine("Datum und Uhrzeit: " + DateTime.Now);
    Console.ResetColor();
}
```

Programmausgabe auf der Konsole:

```
HALLÖCHEN!
©SLOG, Wien 2021
Datum und Uhrzeit: 27.02.2021 21:21:20
Gib deine Körpergröße in Zentimeter ein: 163
Du bist 5,35 Fuß groß.
©SLOG, Wien 2021
Datum und Uhrzeit: 27.02.2021 21:21:43
```

```
AUF WIEDERSEHEN!
```

Methoden mit mehr als einem Inputparameter

Eine Methode kann zum Glück eine beliebige Anzahl von Inputparameter haben, nämlich gar keine, einen oder auch viel mehr. Die Inputparameter werden einfach durch Beistriche getrennt.

Wir erweitern unsere Methode um die Möglichkeit einen 277-Hz-Ton[2] für eine bestimmte Dauer über den Lautsprecher auszugeben. Das ist über eine Methode `Console.Beep(int,int)`[3] möglich, der man die Frequenz und die Dauer in Millisekunden übergibt.

In unserem Programm wird im Main für die Dauer des Tons einmal 0.3 und einmal 0.5 übergeben.

Genauso wie mit dem Inputparameter `farbe`, wird auch der Inputparameter `sekunden` während der Programmausführung mit dem Wert des Aufrufs initialisiert.

```
static void CopyrightMeldungAusgeben(ConsoleColor farbe, double sekunden)
{
    Console.ForegroundColor = farbe;
    Console.WriteLine("\u00A9SLOG, Wien 2021"); // U+00A9 ist das Copyrightzeichen
    Console.WriteLine("Datum und Uhrzeit: " + DateTime.Now);
    Console.Beep(277 /* in Hz (277 ist die Note C#) ;-) */, (int)(sekunden * 1000));
    Console.ResetColor();
}

static void Main(string[] args)
{
    Console.OutputEncoding = Encoding.Unicode;

    // Willkommensmeldung
    Console.WriteLine("HALLÖCHEN!");
    CopyrightMeldungAusgeben(ConsoleColor.Yellow, 0.3);

    // Eingabe und Berechnung
    Console.Write("Gib deine Körpergröße in Zentimeter ein: ");
    int cm = int.Parse(Console.ReadLine());
    double feet = Math.Round(cm * 3.28084e-2, 2); // gerundet auf zwei Stellen
    Console.WriteLine("Du bist " + feet + " Fuß groß.");

    // Verabschiedung
    CopyrightMeldungAusgeben(ConsoleColor.Red, 0.5);
    Console.WriteLine("AUF WIEDERSEHEN!");

    Console.ReadKey();
}
```

Programmausgabe:

```
HALLÖCHEN!
©SLOG, Wien 2021
Datum und Uhrzeit: 27.02.2021 21:24:56 C#-Beep!!!
Gib deine Körpergröße in Zentimeter ein: 163
Du bist 5,35 Fuß groß.
©SLOG, Wien 2021
Datum und Uhrzeit: 27.02.2021 21:25:02 C#-Beep!!!!!
```

[2] 277,18 Hz entspricht genau der Note **C♯** (siehe https://en.wikipedia.org/wiki/C%E2%99%AF_(musical_note)

[3] https://msdn.microsoft.com/de-de/library/4fe3hdb1(v=vs.110).aspx

AUF WIEDERSEHEN!

Man kann eine Methode auch so aufrufen, dass die Werte für die Inputparameter mit dem Namen und in beliebiger Reihenfolge angegeben sind.

In unserem Beispiel z. B.:

```
CopyrightMeldungAusgeben(sekunden: 0.3, farbe: ConsoleColor.Yellow);
```

Hinweis: Früher hat man Methoden die keinen Rückgabewert hatten, auch oft als Prozeduren bezeichnet[4].

[4] In Sprachen Pascal oder Modula 2 wurden Methoden ohne Rückgabewert mit dem Schlüsselwort `procedure` definiert, während alle anderen Methoden mit dem Schlüsselwort `function` definiert wurden. Den Begriff *Methode* gibt es erst seit der Entwicklung von objektorientierten Sprachen.

Methoden mit einem Rückgabetyp

Methoden können über Inputparameter Werte erhalten und diese verwenden.

Methoden können aber auch über mehrere Arten Werte zurückliefern oder in übergebenen Parametern Werte so verändern, dass sie an der Programmstelle des Methodenaufrufs verwendet werden können.

Die häufigste Art, dass Methoden etwas zurückgeben ist mittels eines Rückgabetyps. Und das geht relativ einfach. Statt `void` gibt man links vom Methodennamen den Datentyp an, den die Methode zurückgeben soll.

Beispiel: Von einigen Wörtern sollen die Selbstlaute gezählt werden.

```
static int SelbstlauteZählen(string s)
{
    if (s == "")
       // Optimierung: Leere Strings haben nie Selbstlaute.
       return 0;

    int zähler = 0;

    for (int i = 0; i < s.Length; i++)
    {
        string z = s.Substring(i, 1);

        if ("aAeEiIoOuU".Contains(z))
        {
            zähler++;
        }
    }

    return zähler;
}

static void Main(string[] args)
{
    string[] wörter = new string[4];
    wörter[0] = "Kürbiscremesuppe";
    wörter[1] = "Erdäpfelpüree";
    wörter[2] = "Schokomuffins";
    wörter[3] = "";

    for (int i = 0; i < wörter.Length; i++)
    {
        string wort = wörter[i];
        int anzahl = SelbstlauteZählen(wort);
        Console.WriteLine("<" + wort + "> enthält " + anzahl + " Selbstlaute.");
    }

    Console.ReadKey();
}
```

Programmausgabe auf der Konsole:

```
<Kürbiscremesuppe> enthält 5 Selbstlaute.
<Erdäpfelpüree> enthält 4 Selbstlaute.
<Muffins> enthält 2 Selbstlaute.
<> enthält 0 Selbstlaute.
```

Wenn eine Methode mehr als einen Wert zurückgeben muss, kann man diese Werte in einer Struct speichern und diese zurückgeben.

Wie das untere Programm zeigt, ist das allerdings eher aufwendig und eher mühselig. Es gibt eine weitaus einfachere Möglichkeit, dass eine Methode mehrere Werte an den Aufruf zurückgibt.

Beispiel: Von einigen Wörtern sollen die Selbstlaute und Umlaute gezählt werden.

```
struct BuchstabenAnzahl
{
    public int AnzahlSelbstlaute;
    public int AnzahlUmlaute;
}

static BuchstabenAnzahl Zählen(string s)
{
    BuchstabenAnzahl zähler;
    zähler.AnzahlSelbstlaute = 0;
    zähler.AnzahlUmlaute = 0;

    if (s == "")
        // Optimierung: Leere Strings haben nie Selbstlaute.
        return zähler;

    for (int i = 0; i < s.Length; i++)
    {
        string z = s.Substring(i, 1);

        if ("aAeEiIoOuU".Contains(z))
        {
            zähler.AnzahlSelbstlaute++;
        }

        if ( "äÄöÖüÜ".Contains(z) )
        {
            zähler.AnzahlUmlaute++;
        }
    }

    return zähler;
}
```

```
static void Main(string[] args)
{
    string[] wörter = new string[]
    {
        "Kürbiscremesuppe",
        "Erdäpfelpüree",
        "Muffins",
        ""
    };

    for (int i = 0; i < wörter.Length; i++)
    {
        string wort = wörter[i];
        BuchstabenAnzahl anz = Zählen(wort);
        Console.WriteLine("<" + wort + "> enthält "
            + anz.AnzahlSelbstlaute + " Selbstlaute und "
            + anz.AnzahlUmlaute + " Umlaute.");
    }
```

Programmausgabe auf der Konsole:

```
<Kürbiscremesuppe> enthält 5 Selbstlaute und 1 Umlaute.
<Erdäpfelpüree> enthält 4 Selbstlaute und 2 Umlaute.
<Muffins> enthält 2 Selbstlaute und 0 Umlaute.
<> enthält 0 Selbstlaute und 0 Umlaute.
```

Methoden mit Outputparameter

Eine Methode hat nicht nur Inputparameter, sondern auch Outputparameter.

Outputparameter sind in einer Methode Variablen, die befüllt werden. Sie werden in der Parameterliste durch das Wort `out` markiert.

Beispiel: Von einigen Wörtern sollen die Selbstlaute und Umlaute gezählt werden. Dafür werden zwei Outputparameter, `anzahlSelbstlaute` und `anzahlUmlaute`, deklariert und in der Methode auf berechnet. Beim Aufruf können dann die Werte außen verwendet werden.

```
static void Zählen(string s, out int anzahlSelbstlaute, out int anzahlUmlaute )
{
    anzahlSelbstlaute = anzahlUmlaute = 0;

    if (s == "")
        // Optimierung: Leere Strings haben nie Selbstlaute.
        return;

    for (int i = 0; i < s.Length; i++)
    {
        string z = s.Substring(i, 1);

        if ("aAeEiIoOuU".Contains(z))
        {
            anzahlSelbstlaute++;
        }

        if ( "äÄöÖüÜ".Contains(z) )
        {
            anzahlUmlaute++;
        }
    }
}

static void Main(string[] args)
{
    string[] wörter = new string[]
    {
        "Kürbiscremesuppe",
        "Erdäpfelpüree",
        "Muffins",
        ""
    };

    for (int i = 0; i < wörter.Length; i++)
    {
        string wort = wörter[i];
        int anzSelbstlaute, anzUmlaute;
        Zählen(wort, out anzSelbstlaute, out anzUmlaute);

        Console.WriteLine("<" + wort + "> enthält "
            + anzSelbstlaute + " Selbstlaute und "
            + anzUmlaute + " Umlaute.");
    }

    Console.ReadKey();
}
```

WICHTIG: Auch beim Aufruf muss das Schlüsselwort `out` angegeben werden!

Programmausgabe auf der Konsole:

```
<Kürbiscremesuppe> enthält 5 Selbstlaute und 1 Umlaute.
<Erdäpfelpüree> enthält 4 Selbstlaute und 2 Umlaute.
<Muffins> enthält 2 Selbstlaute und 0 Umlaute.
<> enthält 0 Selbstlaute und 0 Umlaute.
```