## Methoden in C#

Bisher sprachen wir v. a. darüber, wie man den Ablauf eines Programms durch Verzweigungen und Schleifen steuert. Aber schon in der Einführungsstunde zur Objektorientierten Programmierung sahen wir, dass jedes Objekt Eigenschaften und Methoden bzw. Operationen besitzt. In Methoden kann jedes Objekt entsprechend der Werte seiner Eigenschaften handeln. Methoden ähneln den Funktionen aus der Mathematik.

Wir haben bereits vorhandene Methoden verwendet, z. B. `Parse` oder `ToString();` oder vorgegebene „Methodengerüste“ bei Ereignisbehandlungen im Leben erfüllt.

Heute und in den nächsten Stunden geht es darum, eigene Methoden zu erstellen. Wir beginnen mit dem einfachsten Fall:

### Methoden ohne Parameter

Alle Methoden werden immer **innerhalb einer Klasse** erstellt. Methoden ohne Parameter haben den folgenden Aufbau:

```
<Rückgabetyp> <Methodenname>()                    Methodenkopf
{
  // Code, der in der Methode ausgeführt wird     Methodenrumpf
}
```

Unter `<Rückgabetyp>` einer Methode versteht man den Datentyp, den die Methode zurückliefern soll. In der Mathematik liefert z. B. die Funktion sin(x) ein Ergebnis vom Typ double zurück. Wenn man eine Methode keinen Wert zurückliefern soll, schreibt man als Typ „`void`“.

Der `<Methodenname>` ist ein Bezeichner (Regeln: siehe früher), unter dem die Methode von C# erkannt werden soll.

Beispiel 1: Formular mit Button button1

```
1  void sageHallo(){
2    for(int i=1;i<=3;i++){
3      MessageBox.Show("Hallo!");
4    }
5  }
6
7  void Button1Click(object sender, System.EventArgs e){
8    MessageBox.Show("Button 1 wurde geklickt!");
9    sageHallo();
10   MessageBox.Show("wieder zurück in Button1Click");
11 }
```

Schauen wir uns das Beispiel ein wenig genauer an:

- In den Zeilen 1 bis 5 befindet sich unsere eigene Methode mit dem Namen `sageHallo` mit keinem Parameter und keinem Rückgabewert (erkennbar an `void`). Eine Methode ohne Parameter dient dazu, ein bestimmtes Unterprogramm, das mehrmals in genau der gleichen Weise ablaufen soll, nur einmal programmieren bzw. schreiben zu müssen.
- In der Methode selbst wird dreimal ein Meldungsfenster mit "Hallo" angezeigt.
- In den Zeilen 7 bis 11 befindet sich die Ereignisbehandlung für `Button1Click`, die aufgerufen wird, wenn Button1 gedrückt wird. Zunächst wird ein Meldungsfenster mit dem Text "Button 1 wurde geklickt" ausgegeben.
- In der Zeile 9 wird der Ablauf der Ereignisbehandlung gestoppt und unsere eigene Methode sageHallo aufgerufen. Eine Methode ohne Parameter wird aufgerufen, in dem man ihren Namen und direkt anschließend runde Klammer schreibt. Wenn ihre Abarbeitung beendet ist, wird die Ereignisbehandlung direkt nach dem Methodenaufruf fortgeführt und ein weiteres Meldungsfenster ausgegeben.

 

Beispiel 2: Formular mit Button button1

```
1  String liefereUhrzeit(){
2       String d;
3       d = DateTime.Now.ToString();
4       return d;
5  }
6
7  void Button1Click(object sender, System.EventArgs e){
8       String s=liefereUhrzeit();
9       MessageBox.Show(s);
10 }
```

- In diesem Beispiel heißt unsere eigene Methode liefereUhrzeit und gibt einen Wert vom Typ String zurück, was man in Zeile 1 erkennt.
- In Zeile 3 wird das aktuelle Datum und die Uhrzeit bestimmt und im String d gespeichert.
- Hinter dem **`return`** in Zeile 4 steht der Wert, der zurückgegeben wird. Statt der Variablen d kann auch ein beliebiger Ausdruck stehen, der aber den im Methodenkopf angegebenen Datentyp ansprechen muss. Wichtig ist, dass **`return`** die letzte in der Methode ausgeführte Anweisung sein muss!

Aufgabe 1:

a) Kompiliere das Programm. Was wird ausgegeben?
b) Lösche die Zeile 4 und versuche das Programm zu kompilieren. Was passiert?
c) Tippe Zeile 4 wieder ein und füge direkt darunter eine weitere Zeile mit `MessageBox.Show("Ausgabe!");` ein. Kompiliere das Programm und führe es aus. Überprüfe außerdem die Meldungen beim Kompilieren. Notiere, was du liest.
d) Entferne die zusätzliche Zeile aus dem c)-Teil und ändere die Methode so, dass als Ausgabe z. B. "Jetzt ist: 23.2.2005 14:00:10" zurückgeliefert wird.
e) Was passiert, wenn du in der Methode `liefereUhrzeit` die Variable d in s umbenennst? *Wichtig: Bitte jedes Auftreten von d ersetzen!*
f) Wie kannst du die Methode so kurz wie möglich formulieren?

### Regeln für Methoden:

- **Block-Regel:**

| | |
|---|---|
| Block: | Bereich zwischen **{** und **}** |
| Gültigkeitsbereich: | Eine Konstante bzw. Variable ist in dem Programmblock gültig, in dem sie vereinbart wurde, sowie in allen untergeordneten Blöcken. |
| Globaler Name/ Instanzvariable oder Datenfeld: | Eine Variable, die außerhalb von Methoden definiert wird. Sie hat in der gesamten Klasse Gültigkeit. Global heißt „in allen untergeordneten Methoden und Programmteilen bekannt und gültig“. In der Objektorientierten Programmierung spricht man lieber von Instanzvariable oder Datenfeld. |
| Lokaler Name: | Eine Konstante bzw. Variable, die in einem Block vereinbart wurde. Diese hat nur innerhalb dieses Blockes ihre Gültigkeit. Wir kennen dieses Verhalten von Schleifen. Wenn innerhalb des Schleifenrumpfes eine Variable definiert wird, kann man außerhalb dieses Schleifenblockes nicht mehr auf sie zugreifen. |

- **Ausblenden-Regel:** Treten in einem Programm globale und lokale Größen mit gleichem Namen auf, so gilt innerhalb einer Methode nur der **lokale Name**. Die globale Größe wird vorübergehend ausgeblendet (ignoriert). Nach Verlassen der Methode hat die globale Größe noch ihren vorherigen Wert.
- Methoden können **nicht ineinander geschachtelt** werden.

 