



Benutzen einer Methode in C#

Eine Methode in C# ist ein Codeblock, der eine bestimmte Aufgabe ausführt. Eine Methode kann Werte empfangen, einen Prozess durchführen und dann einen Wert zurückgeben. Hier ist eine Schritt-für-Schritt-Anleitung, wie eine Methode in C# funktioniert:

1. **Deklaration:** Zunächst müssen Sie die Methode deklarieren. Eine Methode Deklaration teilt dem Compiler mit, wie die Methode aussieht. Eine Methode Deklaration hat die folgenden Teile:
 - **Zugriffsmodifikator:** Dies legt fest, von wo die Methode zugänglich ist. Beispielsweise kann eine Methode **public** sein, was bedeutet, dass sie von jeder anderen Methode in jedem anderen Objekt oder Klasse aufgerufen werden kann. Andere Zugriffsmodifikatoren sind **private**, **protected**, **internal**, und **protected internal**.
 - **Rückgabetyt:** Der Datentyp des Wertes, den die Methode zurückgeben wird. Wenn die Methode keinen Wert zurückgibt, ist der Rückgabetyt **void**.
 - **Methodenname:** Der Name der Methode. Dies ist der Name, den Sie verwenden, um die Methode aufzurufen.
 - **Parameterliste:** Eine Liste von Variablen, die an die Methode übergeben werden. Wenn die Methode keine Parameter erfordert, ist diese Liste leer.

Hier ist ein Beispiel für eine Methode Deklaration in C#:

```
public int AddNumbers(int a, int b)
```

2. In diesem Beispiel ist **public** der Zugriffsmodifikator, **int** ist der Rückgabetyt, **AddNumbers** ist der Methodenname, und **(int a, int b)** ist die Parameterliste.
3. **Definition:** Nachdem Sie die Methode deklariert haben, müssen Sie die Methode definieren, d. h. den Code schreiben, der ausgeführt wird, wenn die Methode aufgerufen wird. Die Methode Definition besteht aus dem Methodenkopf (die Methode Deklaration) und dem Methodenkörper (ein Block von Anweisungen zwischen geschweiften Klammern { }).

Hier ist ein Beispiel für eine Methode Definition in C#:

```
public int AddNumbers(int a, int b)
{
    int result = a + b;
    return result;
}
```

In diesem Beispiel besteht der Methodenkörper aus zwei Anweisungen: Eine, die zwei Zahlen addiert und das Ergebnis in einer Variablen speichert, und eine **return** Anweisung, die das Ergebnis zurückgibt.



4. **Aufruf:** Nachdem Sie eine Methode definiert haben, können Sie sie aufrufen, d. h. den Code im Methodenkörper ausführen. Um eine Methode aufzurufen, verwenden Sie den Methodennamen gefolgt von einer Liste von Argumenten in Klammern. Die Argumente müssen mit den Parametern in der Methode Deklaration übereinstimmen.

Hier ist ein Beispiel für einen Methodenaufruf in C#:

```
int sum = AddNumbers(5, 10);
```

In diesem Beispiel wird die AddNumbers Methode mit den Argumenten 5 und 10 aufgerufen. Das Ergebnis, das von der Methode zurückgegeben wird, wird in der Variablen sum gespeichert.

Methoden dienen mehreren Zwecken in der Programmierung:

1. **Wiederverwendbarkeit von Code:** Wenn Sie denselben Code an mehreren Stellen in Ihrem Programm verwenden müssen, können Sie diesen Code in eine Methode schreiben und dann diese Methode jedes Mal aufrufen, wenn Sie diesen spezifischen Code ausführen müssen. Dies hilft, Duplikationen von Code zu vermeiden und erleichtert Ihr Programm zu verstehen und zu warten.
2. **Modularität:** Methoden helfen dabei, den Code in kleinere, handhabbare Teile (Module) zu unterteilen. Jede Methode führt eine spezifische Aufgabe aus. Dies macht es einfacher, den Code zu verstehen, zu testen und zu debuggen, da Sie sich auf eine Aufgabe zur Zeit konzentrieren können.
3. **Wartbarkeit:** Da Methoden den Code in kleinere Teile unterteilen, ist es einfacher, Änderungen am Programm vorzunehmen. Wenn Sie eine Änderung an einer spezifischen Funktion Ihres Programms vornehmen müssen, müssen Sie nur den Code innerhalb der entsprechenden Methode ändern, ohne den Rest des Programms zu beeinflussen.
4. **Verständlichkeit:** Ein gut strukturiertes Programm, das Methoden verwendet, um verschiedene Aufgaben zu erledigen, ist leichter zu verstehen, da es klar ist, welche Teile des Codes welche Aufgaben erledigen.
5. **Wiederverwendbarkeit:** Methoden ermöglichen es, Code zu erstellen, der in verschiedenen Teilen eines Programms oder sogar in anderen Programmen wiederverwendet werden kann. Dies hilft, die Entwicklung zu beschleunigen, da Sie nicht den gleichen Code mehrmals schreiben müssen.

Zum Beispiel, wenn Sie eine Methode schreiben, um zwei Zahlen zu addieren, können Sie diese Methode jedes Mal verwenden, wenn Sie Zahlen in Ihrem Programm addieren müssen, anstatt den Code zum Addieren von Zahlen jedes Mal neu zu schreiben.