ZUSAMMENFASSUNG Fallunterscheidungen

WAS SIND IF-ABFRAGEN?

- Mit if-Abfragen (If-Statements) kann man Codeblöcke zu bestimmten Bedingungen ausführen
- Sie ermöglichen eine Kontrolle über die Ausführung von Code

DER AUFBAU EINER IF-ABFRAGE

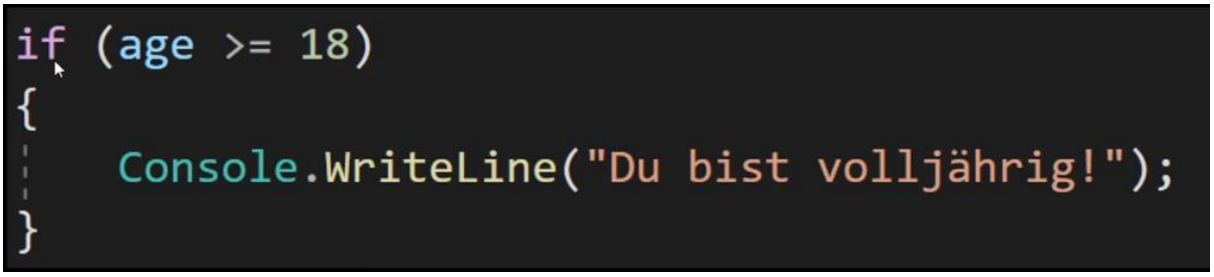

Abbildung 1, Aufbau einer if-Abfrage

- Eine if-Anweisung wird mit dem Schlüsselwort **„if“** eingeleitet
- Neben dem **„if“** steht ein Klammerpaar, welches eine **Bedingung** definiert
- Wird die Bedingung erfüllt, dann wird der zur if-Anweisung gehörende Codeblock ausgeführt

ALTERNATIVE CODEBLÖCKE

Wenn die Bedingung nicht erfüllt wird, kann man alternative Bedingungen abfragen oder einfach einen alternativen Codeblock ohne weitere Bedingung ausführen:

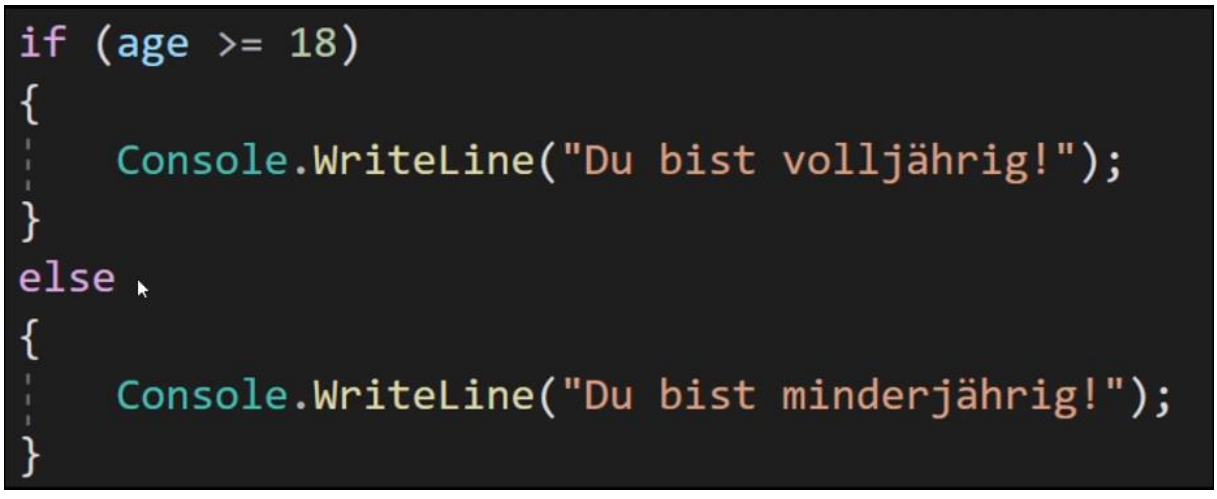

Abbildung 2, else-Block bei einem if-Statement

IF-ABFRAGEN IN DER PRAXIS

Im folgenden if-Statement wird überprüft, ob das Alter des Nutzers **größer oder gleich** 18 ist. In einem alternativen Codeblock wird abgefragt, ob die Eltern dabei sind. Anschließend wird für die Bedingung, die erfüllt wird, ein Satz ausgegeben.

```
static void Main(string[] args)
{
    int age = 14;
    bool withParents = false;

    if (age >= 18)
    {
        Console.WriteLine("Du darfst den Film sehen!");
    }
    else if (withParents == true)
    {
        Console.WriteLine("Du darfst den Film dank deiner Eltern sehen!");
    }
    else
    {
        Console.WriteLine("Du darfst den Film nicht sehen!");
    }
    Console.ReadKey();
}
```

Abbildung 3, Mehrere Bedingungen überprüfen

Abbildung 4, Output

BOOLESCHE AUSDRÜCKE UND OPERATOREN

Was ist ein Ausdruck?

- Ein Ausdruck ist eine Rechnung, die ein Ergebnis liefert
- Beispiele für mathematische Ausdrücke sind:
  (2 + 4) -> 6
  (3 * (3 + 5)) -> 24
  (32 * 2) -> 64
  (Radius * Radius * PI) -> Kreisfläche

Was ist ein boolescher Ausdruck?

- Ein boolescher Ausdruck kann nur die Werte **„True“** oder **„False“** als Ergebnisse liefern
- Beispiele:
  (5 > 3) -> **true**
  (10 == 4) -> **false**
  (2 != 3) -> **true**

```
bool Bedingung1 = (5 > 3);
bool Bedingung2 = (10 == 4);
bool Bedingung3 = (2 != 3);
```

Abbildung 5, boolesche Ausdrücke

Alle booleschen Operatoren

Vergleichsoperatoren

Operator	Zweck	Beispiel
<	Kleiner als	(2 < 4)
>	Größer als	(3 > 1)
<=	Kleiner/Gleich	(3 <= 3)
>=	Größer/Gleich	(2 >= 1)
==	Gleich	(2 == 2)
!=	Ungleich	(4 != 8)

Logische-/Verknüpfungsoperatoren

Operator	Zweck	Beispiel
&&	Und	((20 > 3) && (10 == 10))
||	Oder	((10 > 20) || (2 != 3))
!	Nicht	! ((20 > 3) && (10 == 10))

Abbildung 6, alle booleschen Operatoren

BOOLESCHE AUSDRÜCKE IN DER PRAXIS

Im nachfolgenden Beispiel überprüfen wir, ob sich ein Bewerber für einen Job eignet:

```
{
    int alter = 18;
    bool mobil = false;
    bool qualifiziert = true;
    bool testBestanden = false;
    bool schüchtern = false;

    if ((alter >= 18) && (mobil == true) && (qualifiziert == true || testBestanden == true) && schüchtern == false)
    {
        Console.WriteLine("Du bekommst den Job!");
    }
    else
    {
        Console.WriteLine("Du bekommst den Job leider nicht!");
    }

    Console.ReadKey();
}
```

Abbildung 7, boolesche Ausdrücke in der Praxis

Abbildung 8, Output

Hinweis:

Schreibweise	**Alternative Schreibweise**
if (mobil == true)	-> if (mobil)
if (schüchtern == false)	-> if (!schüchtern)

WAS SIND SWITCH-BLÖCKE?

- Switch-Blöcke sind eine weitere Art der Fallunterscheidung in C#
- Anders als bei if-Abfragen prüft man keine Bedingung, sondern eine Variable auf eine Reihe von Werten
- Jedem Wert wird dabei ein „Case“ zugewiesen, welcher ausgeführt wird, wenn die Variable den dazugehörenden Wert beinhaltet
- if-Abfragen -> prüfen eine Bedingung
- Switch-Blöcke -> prüfen den Wert einer Variable auf konstante Werte

DER AUFBAU EINES SWITCH-BLOCKS

```
switch(day)
{
    case 1:
        Console.WriteLine("Es ist Montag");
        break;
    case 2:
        Console.WriteLine("Es ist Dienstag");
        break;
    case 3:
        Console.WriteLine("Es ist Mittwoch");
        break;
    case 4:
        Console.WriteLine("Es ist Donnerstag");
        break;
    case 5:
        Console.WriteLine("Es ist Freitag");
        break;
    default:
        Console.WriteLine("Ungültiger Wert");
        break;
}
```

Abbildung 9, Aufbau eines Switch-Blocks

- Zuerst schreibt man das Schlüsselwort **„switch“**
- Danach kommt ein Klammerpaar, welches eine Variable enthält, deren Wert man gegen andere Werte prüfen möchte
- Im Switch-Block werden **„cases“** definiert, welche einem konstanten Wert zugeordnet sind
- Hat die Variable denselben Wert wie ein **„case“**, wird der Code vom case ausgeführt
- Mit dem Schlüsselwort **„break“** springt man aus dem Switch-Block heraus

SWITCH-BLÖCKE IN DER PRAXIS

Auf der nachfolgenden Abbildung sehen wir ein Switch-Statement, bei welchem der Benutzer etwas eingeben kann. Daraufhin wird überprüft, ob die Eingabe des Nutzers einem der „cases“ entspricht. Gibt er zum Beispiel das Wort „SayHello“ ein, so wird ihm der Satz „Hallo!“ ausgegeben.

```
{
    Console.Write("Command: ");
    string command = Console.ReadLine();

    switch(command)
    {
        case "SayHello":
            Console.WriteLine("Hallo!");
            break;

        case "SayGoodbye":
            Console.WriteLine("Auf Wiedersehen!");
                break;

        case "Smile":
            Console.WriteLine(":)");
            break;

        default:
            Console.WriteLine("Ungültiger Befehl!");
            break;
    }
    Console.ReadKey();
}
```

Abbildung 10, Switch-Statement in der Praxis