Zusammenfassung Advanced Programming

Joel von Rotz

19.06.1932

Table of contents

Vergleich C & C#

- Jede Funktion muss zu einer Klasse gehören. Es gibt keine "nackten" Funktionen

Datentypen

String

Strings werden mit dem folgender Deklaration `inhalt"`

> **!** Important
>
> Strings können nicht verändert werden -> sind **read-only**

```
string s = "Hallo Welt";

s[1] = 'A'; // ERROR
```

Parameter in String einfügen

Parameter/variablen können in Strings direkt eingefügt werden.

Bildschirmausgabe

Variante 1 - C Style:

```
Console.WriteLine("The sum of {0} and {1} is {2}",a,b,result);
```

Variante 2 - C# Style:

```
Console.WriteLine("The sum of" + a + "and" + b + "is" + result);
```

Variante 3 - new C# Style:

```
Console.WriteLine($"The sum of {a} and {b} is {result}");
```

Overloading

Konstruktor Overloading

```
class Point {
  private int pos_x;
  private int pos_y;

  public Point(int x, int y) {
    this.pos_x = x;
    this.pos_y = y;
  }

  public Point() : this(0,0) {}
}
```

Mit `this` nach dem Konstruktor (unterteilt mit `:`) kann der Aufruf auf einen anderen Konstruktor weitergeleitet werden. Der Inhalt des vorherigen Konstruktors wird erst nach dem Ablauf des `this`-Konstruktors (im Beispiel `Point(int x, int y)`).

Konstruktor Aufruf-Reihenfolge

```
using System;

class Point {
  private int pos_x;
  private int pos_y;

  public Point(int x, int y) {
    this.pos_x = x;
    this.pos_y = y;
    Console.WriteLine($"Point {this.pos_x}, {this.pos_y}");
  }

  public Point(int x) : this(x, 0) {
    Console.WriteLine("x-only");
  }

  public Point() : this(0,0) {}
  Console.WriteLine("no value");
}
```

Wird nun `Point(4)` aufgerufen erhält man folgendes auf der Konsole

```
Point 4, 0
x-only
```

Default Parameter (implizit Overloading)

Funktion

`out`

Notes

Overflows Integer

Im folgenden Code wird eine Variable `i` mit dem maximalen Wert eines `int` geladen und folgend inkrementiert.

```
int i = int.MaxValue;
i++;
```

Wird aber dies direkt in der Initialisierung eingebettet (`...+ 1`), ruft der Compiler aus, da er den Overflow erkennt.

```
int i = int.MaxValue + 1; // COMPILE-FEHLER
i++;
```

> **Danger**
>
> Dieser Overflow-Fehler gilt nur bei **konstanten** Werten bei der Initialisierung. Wird eine separate Variable mit dem Maximalwert initialisierit und an `i` hinzuaddiert, gibt es keinen Fehler.
>
> ```
> int k = int.MaxValue;
> int i = k + 1; // KEIN Fehler
> ```