Java!

C#?

Einführung

Jasmin Hellebronth

2016-04-14

Compiler und Runtime

Eine .cs-Datei wird vom Compiler (csc) in IL-Code übersetzt und als .exe-Datei gespeichert.
Der IL-Code wird zur Laufzeit vom JIT-Compiler in Maschinencode übersetzt.

IL = Intermediate Language: prozessorunabhängige Zwischensprache
JIT = Just-In-Time-Compiler: Teil der Common Language Runtime

csc /out : <Dateiname>.exe <Dateiname>.cs

Hello World

```
using System;
class MyClass
{
        public static void Main()
        {
                Console.WriteLine(„Hello World“);
        }
}
```

Die Namen der Klasse und der Datei müssen nicht gleich sein.
Eine Datei darf mehr als eine Klasse beinhalten.

Namensräume

```
using SC = System.Console;
class MyClass
{
        public static void Main()
        {
                SC.WriteLine(„Hello World“);
        }
}
```

Namensräume sind unabhängig vom Dateipfad.

Namensräume

```
class MyClass {
        public static void Main() {
                NS.MyClass.Bye();
        }
}
namespace NS {
        using SC = System.Console;
        class MyClass {
                public static void Bye() {
                        SC.WriteLine("Bye bye!");
                }
        }
}
```

Namenskonvetionen

Klassen und Methoden in PascalCasing benennen.
Variablen und Konstanten in camelCase benennen.

Typen

byte	0 ... 255
sbyte	–128 ... 127
short	-2^{15} ... $2^{15}-1$
ushort	0 ... 65535
int	-2^{31} ... $2^{31}-1$
uint	0 ... $2^{32}-1$
long	-2^{63} ... $2^{63}-1$
ulong	0 ... $2^{64}-1$
float	$1{,}4 * 10^{-45}$ bis $3{,}4 * 10^{38}$
double	$5{,}0 * 10^{-324}$ bis $1{,}7 * 10^{308}$
decimal	+/–79E27 ohne Dezimalpunktangabe; +/–7.9E-29, falls 28 Stellen hinter dem Dezimalpunkt angegeben werden. Die kleinste darstellbare Zahl beträgt +/–1.0E-29.
char	Unicode-Zeichen zwischen 0 und 65535
string	ca. 2^{31} Unicode-Zeichen
bool	true oder false

Felder

Primitive Typen können wie Objekte behandelt werden (Autoboxing):

```
int i = 0;
Console.WriteLine(i.ToString());
```

Instanzvariablen und lokale Variablen müssen vor der ersten Verwendung initialisiert werden.

Klassenkonstanten sind const:

```
private const double pi = 3.14;
```

Finale Instanzvariablen sind readonly:

```
private readonly int i;
Konstruktor() { i = 42; }
```

Kontrollstrukturen

if-else
switch-case-default
Bedingter Ausdruck

Schleifen

while
do-while (mindestens ein Durchlauf)
for (bestimmte Anzahl Durchläufe)
foreach:

```
foreach(char c in "Hello World") {
        Console.WriteLine(c);
}
```

goto:

```
Marke:
if(true) goto Marke;
```

Static

Auf statische Methoden kann nicht über eine Instanz zugegriffen werden.

ref und out

Primitive Typen können als Referenz übergeben werden:

```
public static void Main() {
        int i = 0;
        RefMethode(ref i);
        int j;

}
static void RefMethode(ref int i) {
        i++;
}
static void OutMethode(out int j) {
        j = 42;
}
```

params

Mit params kann eine variable Parameterliste übergeben werden:

```
public static void Main() {
        Print("Hello ", "World");
}
public static void Print(params string[] woerter) {
        foreach(string wort in woerter) Console.Write(wort);
}
```

Übergabeparameter mit default

```
public static void Main() {
        Hello();
        Hello("Jay");
}
public static void Hello(string name = "You"){
        Console.WriteLine("Hello " + name);
}
```

Anonyme Klasse

```
public static void Main() {
        var s = new {Name = „Kunz“};
}
```

Es wird implizit eine Klasse mit privaten, finalen Feldern erzeugt.

… und aus der Einführung könnte ein Vorlesung werden …

Vielen Dank fürs Zuhören.