Vergleich zu C#

Florian Gehring

18.06.2020

Hello, World!

```
1  using System;
2  namespace Beispielcode{
3
4  class Hello
5  {
6      static void asdf(String[] args)
7      {
8          Console.WriteLine("Hello, World");
9      }
10 }
11 }
```

Dieses und viele folgende Beispiele: Tour of C#

Geschichte

- Von Microsoft Entwickelt
- „Direkter Konkurent“ zu Java

Typsystem Allgemein

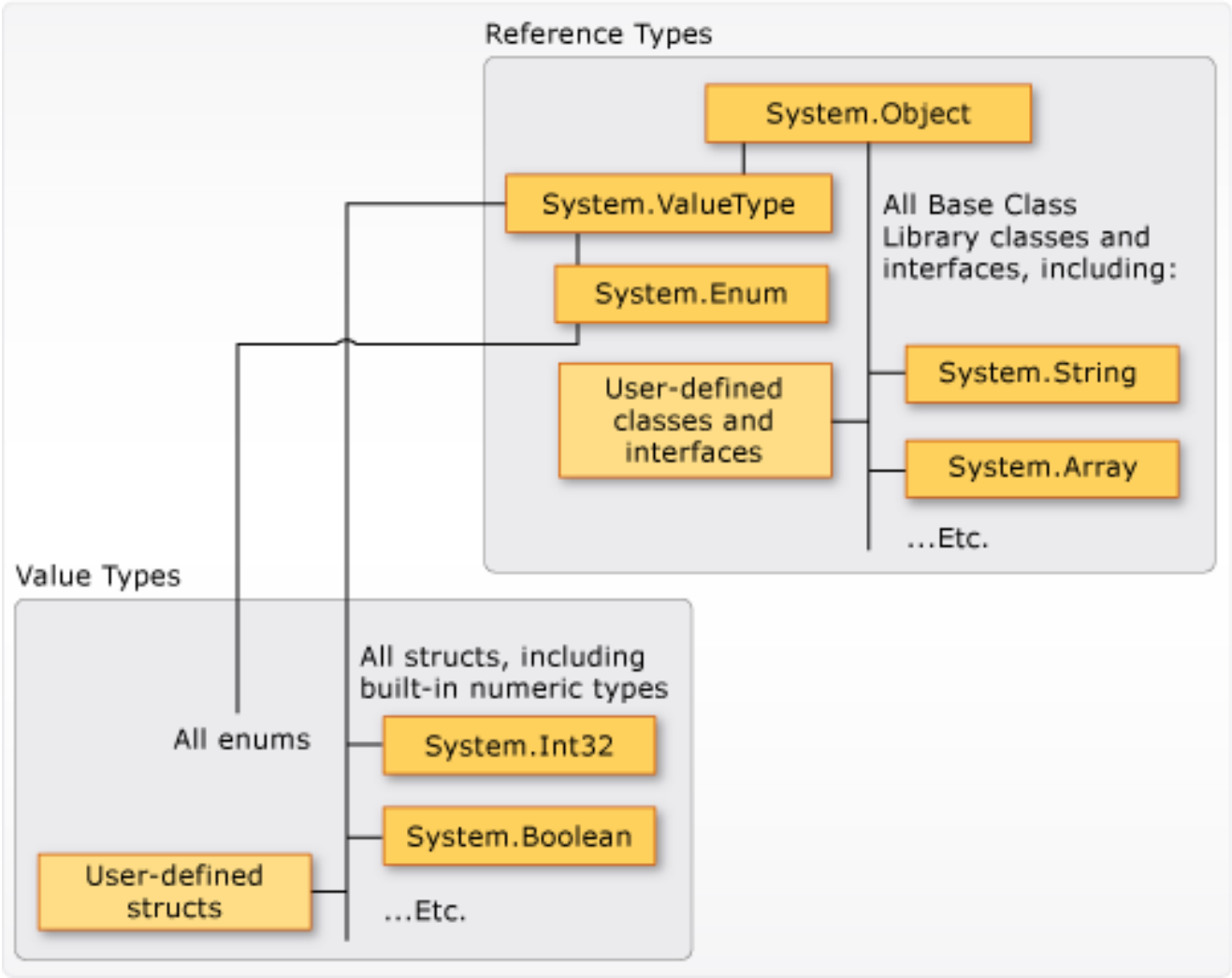

Abbildung: Type System in C# [1]

Allgemeines Typsystem

- Stark Typisiert
  - jede Konstante, Variable und jeder Ausdruck hat einen Typ
- CTS (Common Type System)
  - **Jeder** Typ ist von `System.Object` (`object`) abgeleitet
- Integrierte Typen
  - „Zahlen“, `boolean`, `char`, `object`, `string`
- Referenztypen / Werttypen
  - `System.ValueType`
- Benutzerdefinierte Typen
  - `class`, `enum`, `struct`

Variablen Deklarationen

```
1  using System;
2  using System.Linq;
3          // Declaration only:
4          float temperature;
5          string name;
6          Beispielcode.Hello hello;
7
8
9          // Declaration with initializers (four examples):
10         char firstLetter = 'C';
11         var limit = 3;
12         int[] source = { 0, 1, 2, 3, 4, 5 };
13         var query = from item in source
14                 where item <= limit
```

- var Keyword [4]
- LINQ

Werttypen

```
int i = 2;
// Get the Type
        Type t = i.GetType(); // Methodenaufruf auf int!
        Console.WriteLine(t);
        // Which Methods can be called on this object?
        foreach(var method in t.GetMethods()) {
            Console.Write(method + " ");
        }
        Console.WriteLine();
        // Print Parent classes.
        Type b = t;
        while(b != null) {
            Console.Write(b + " -> ");
            b = b.BaseType;
        }
```

System.Int32
Int32 CompareTo(System.Object) Int32 CompareTo(Int32) ...
System.Int32 -> System.ValueType -> System.Object

Werttypen - Vergleich Java

- In Java: Integer $\neq$ int
- Zwar automatische Konvertierung, aber int ist kein Objekt
- Auskommentierte Zeilen führen zu Fehlern

```java
int i = 2;
// i.getClass();
Object o = i; // <==> Object o = new Integer(i);
System.out.println(o.getClass());
// System.out.println((i instanceof Integer));
System.out.println((o instanceof Integer));
// System.out.println((o instanceof int));
```

Zusammenfassung Typen

Java

- Primitive Typen nicht von `Object` abgeleitet
- Call-by-Value, Call-By-Reference
- Wrapper-Klasse `Integer` für `int`

C#

- Alles (auch `int`) von `object` abgeleitet
- Zahlen, boolean sind „Werttype“
- `int` kann mit `int?` Nullable gemacht werden

Klassen

- Erben implizit von `object`
- Enthalten: `constructors`, `properties`, `indexers`, `events`, `operators` and `destructors`
- `sealed`-Modifier: Für die gesamte Klasse oder einzelne Methoden

Vererbung

```
1  using System;
2  class Base {
3      virtual public void ex1() {
4          Console.WriteLine("Base, example 1");
5      }
6      virtual public void ex2() {
7          Console.WriteLine("Base, example 2");
8      }
9       public void ex3() {
10         Console.WriteLine("Base, example 3");
11     }
12 }
13
14 class Derived : Base{
15
16     new public void ex1() {
17         Console.WriteLine("Derived, example 1");
18     }
19     // sealed: Child classes of Derived can't override ex2
20     sealed override public void ex2() {
21         Console.WriteLine("Derived, example 2, ");
22     }
23
24     //     Forbidden: override public void ex3() ...
25     new public void ex3() {
26         Console.WriteLine("Derived, example 3");
27     }
28 }
```

Vererbung

```
1  class ClassTest {
2      public static void modifierBehavior() {
3          Base trueBase = new Base();
4          Base actuallyDerived = new Derived();
5          Derived trueDerived = new Derived();
6
7          trueBase.ex1(); actuallyDerived.ex1(); trueDerived.ex1();
8          trueBase.ex2(); actuallyDerived.ex2(); trueDerived.ex2();
9          trueBase.ex3(); actuallyDerived.ex3(); trueDerived.ex3();
10
11     }
12 }
```

Base, example 1
Base, example 1
Derived, example 1
Base, example 2
Derived, example 2
Derived, example 2
Base, example 3
Base, example 3
Derived, example 3

Vererbung - Vergleich Java

- Java: Alle Methoden sind virtuell
  - Override wird mit `final` verhindert.
  - Kein Äquivalent zu C# `new`
- Java: `@Override` Dekorator soll Code lesbarer machen
- In C# Ingesamt expliziter als in Java
  - Für interessierte: Interview mit C# Chefdesigner [5]

Quellen

[1] Programming Guide C#", https://docs.microsoft.com/de-de/dotnet/csharp/programming-guide/, 05.06.2020

[2] C# Language Specification, https://docs.microsoft.com/de-de/dotnet/csharp/language-reference/language-specification/introduction, 05.06.2020

[3] Introduction to C#, https://docs.microsoft.com/de-de/dotnet/csharp/language-reference/language-specification/introduction, 06.06.2020

[4] '"What is the equivalent of the C# 'var' keyword in Java?“, https://stackoverflow.com/a/49598148, 05.06.2020

[5] 'Interview with C# Designer', https://www.artima.com/intv/nonvirtual.html, 06.06.2020