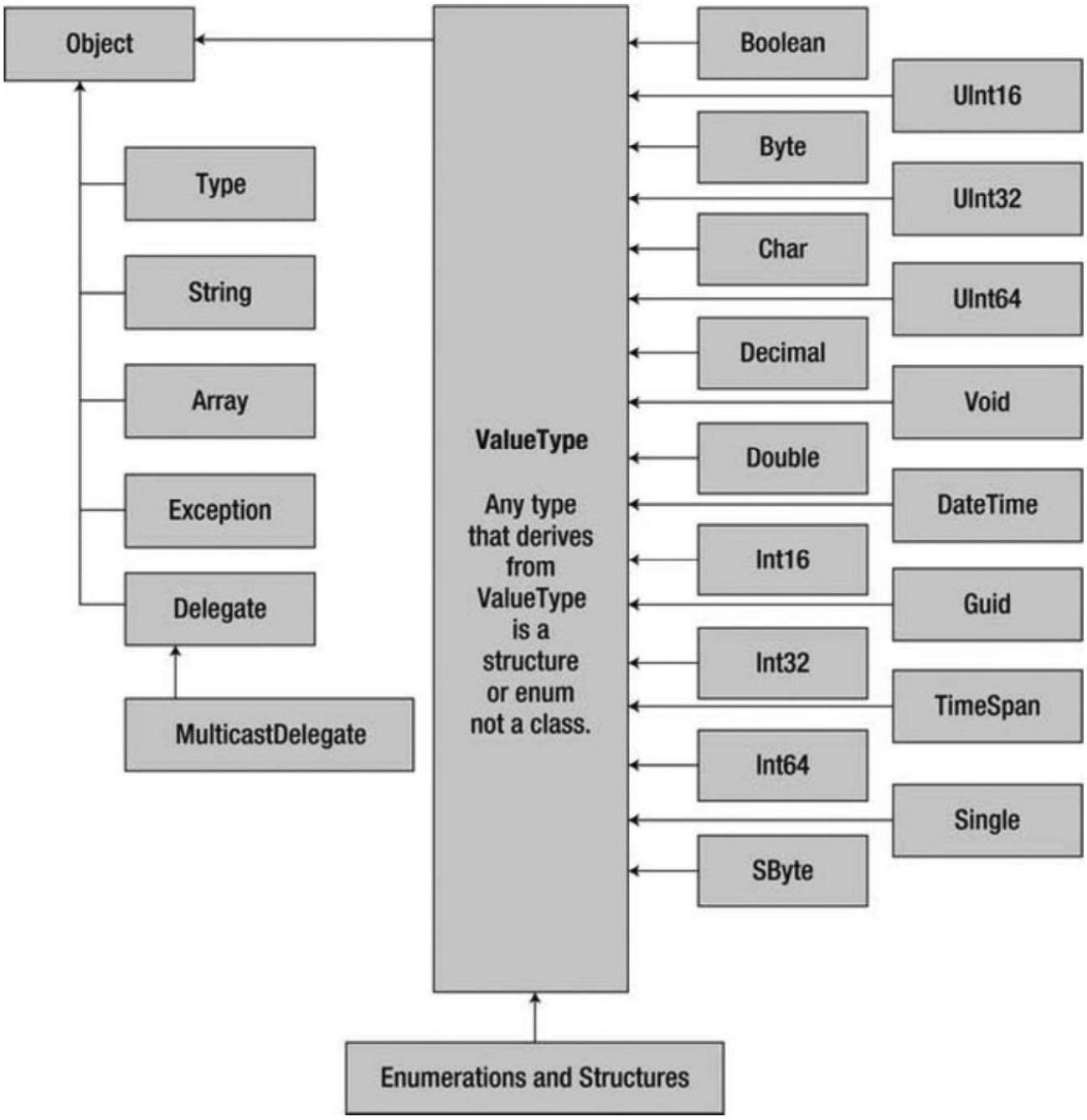
Grundlagen C#

Montag, 12. Dezember 2016 17:12

```
// The topmost class in the .NET universe: System.Object
namespace System
{   public class Object
    {   public Object();  //Konstruktor, wird auch von Konstruktoren abgel. Klassen gerufen
        public virtual Boolean Equals(Object obj); //Aktuelles Objekt und Objekt obj gleich ?
        public static Boolean Equals(Object o1, Object o2); //Objekte o1 und o2 gleich ?
        public virtual int GetHashCode(); //Eindeutiger Wert, override seitens abgel. Klasse
        public Type GetType(); //liefert exakten Typ des Objektes zur Laufzeit
        public virtual String ToString(); //gibt String zurück, der das aktuelle Objekt darstellt
        protected virtual void Finalize(); //Gibt einem Objekt Gelegenheit zum Versuch, Ressourcen
                                           //freizugeben, bevor Objekt vom Garbage Collector freigegeben wird
        protected Object MemberwiseClone(); //Erstellt eine flache Copy des aktuellen Objektes
        public static bool ReferenceEquals(object objA, object objB); //Stellt fest, ob die ange-
        // gebenen Objekte identischen Speicherplatz belegen, d.h. nicht nur gleiche Werte besitzen.
        // Aus ReferenceEquals(o1,o2) == true folgt immer Equals(o1, o2) == true bzw.
        // o1.Equals(o2) == true. Umgekehrt gilt diese Aussage jedoch nicht !
    }
}
```

intrinsische Datentypen

C# Shorthand	CLS Compliant?	System Type	Range	Meaning in Life
bool	Yes	System.Boolean	true or false	Represents truth or falsity
sbyte	No	System.SByte	–128 to 127	Signed 8-bit number
byte	Yes	System.Byte	0 to 255	Unsigned 8-bit number
short	Yes	System.Int16	–32,768 to 32,767	Signed 16-bit number
ushort	No	System.UInt16	0 to 65,535	Unsigned 16-bit number
int	Yes	System.Int32	–2,147,483,648 to 2,147,483,647	Signed 32-bit number
uint	No	System.UInt32	0 to 4,294,967,295	Unsigned 32-bit number
long	Yes	System.Int64	–9,223,372,036,854,775,808 to 9,223,372,036,854,775,807	Signed 64-bit number
ulong	No	System.UInt64	0 to 18,446,744,073,709,551, 615	Unsigned 64-bit number
char	Yes	System.Char	U+0000 to U+ffff	Single 16-bit Unicode character
float	Yes	System.Single	-3.4 10^{38} to +3.4 10^{38}	32-bit floating-point number
double	Yes	System.Double	±5.0 10^{-324} to ±1.7 10^{308}	64-bit floating-point number
decimal	Yes	System.Decimal	(-7.9 x 10^{28} to 7.9 x 10^{28}) / (10^{0} to ^{28})	128-bit signed number
string	Yes	System.String	Limited by system memory	Represents a set of Unicode characters
Object	Yes	System.Object	Can store any data type in an object variable	The base class of all types in the .NET universe



Boxing / Unboxing

boxing: Konvertierung (Einschachteln) eines Wertetyps in ein object (implizit)
unboxing: Extraktion eines Wertetyps aus einem object (explizit)

Beispiel boxing
boxing:

```
Console.WriteLine("12.Equals(23) = {0}", 12.Equals(23));
Console.WriteLine("12.ToString() = {0}", 12.ToString());
Console.WriteLine("12.GetType() = {0}", 12.GetType());
```

Ausgabe:

Ausgabe:
12.Equals(23) =
12.ToString() =
12.GetType() =

unboxing:

```
object o = 123;
int i = (int) o;
```

Member der intrinsischen Datentypen

```
int.MaxValue // liefert größten int
int.MinValue // liefert kleinsten int
double.Epsilon // liefert kleinste darstellbare Zahl

char myChar = 'a';
char.isDigit(myChar) // liefert True wenn myChar eine Zahl ist
char.isLetter(myChar) // liefert True wenn myChar eine Zahl ist
char.isWhiteSpace('Hallo du',5) // steht an 5-ter Stelle ein Leerzeichen ?

Konvertierung von Strings
bool b = bool.Parse(inputString);  // liefert True wenn inputString = "True"
double d = double.Parse(inputString); // convertiert inputString in double
int i = int.Parse(inputString); // convertiert inputString in int
```

Arbeiten mit Strings

String Member	Meaning in Life
`Length`	This property returns the length of the current string.
`Compare()`	This static method compares two strings.
`Contains()`	This method determines whether a string contains a specific substring.
`Equals()`	This method tests whether two string objects contain identical character data.
`Format()`	This static method formats a string using other primitives (e.g., numerical data, other strings) and the `{0}` notation examined earlier in this chapter.
`Insert()`	This method inserts a string within a given string.

`PadLeft()` `PadRight()`	These methods are used to pad a string with some characters.
`Remove()` `Replace()`	Use these methods to receive a copy of a string with modifications (characters removed or replaced).
`Split()`	This method returns a `String` array containing the substrings in this instance that are delimited by elements of a specified `char` array or `string` array.
`Trim()`	This method removes all occurrences of a set of specified characters from the beginning and end of the current string.
`ToUpper()` `ToLower()`	These methods create a copy of the current string in uppercase or lowercase format, respectively.

Beispiel

```
string s = Console.ReadLine(); // z.B. "Hallo Welt"
s = s.Replace("Hallo", "Tschüss");
Console.WriteLine("Modified string: {0}", s); // Tschüss Welt
s = s + " ...";              // string concatenation
Console.WriteLine("Modified string: {0}", s); // Tschüss Welt ...
```

Textformatierung

```
Console.WriteLine("i = {0:F8}", i);
```

allgemein:

{index[,breite][:formatierung]}

Lfd. Nr. des Ausdrucks (von 0 an 0(1) …)

Anzahl Zeichen für Ergebnis der Formatierung,
>0 Rechtsbündig
<0 Linksbündig
wenn fehlt: kürzeste Ganzzahl (Länge)

formatierung

Zeichen	Beschreibung	Defaultergebnis
C, c	Währung	€,$ xx,xx.xx
D, d	dezimal	[-] xxxxxxx
E, e	exponentiell	[-] x.xxxxxxE±xxx
F, f	Festkomma	[-] xxxxxxx.xx
G, g [E, e]	exponentiell	Variabel
N, n	Nummer mit Trennzeichen	[-] xx,xxx.xx
X, x	hexa, ganz	variabel

Escape Characters

Character	Meaning in Life
\'	Inserts a single quote into a string literal.
\"	Inserts a double quote into a string literal.
\\	Inserts a backslash into a string literal. This can be quite helpful when defining file or network paths.
\a	Triggers a system alert (beep). For console programs, this can be an audio clue to the user.
\n	Inserts a new line (on Windows platforms).
\r	Inserts a carriage return.
\t	Inserts a horizontal tab into the string literal.

Widening / Narrowing

Im folgenden Code nimmt die Add Methode nur Argumente vom Typ int, bekommt aber short übergeben. Was passiert ? Welcher Teil des Codes könnte Probleme geben ?

```
public static void Main(string[] args)
{
   short a = 10, b = 10;
   short c = Add(a, b);
   Console.WriteLine("{0} + {1} = {2}", a, b, c);
}

public static int Add(int a, int b)
{
   return a + b;
}
```

Var Keyword

Typen werden automatisch aus der anfänglichen Zuweisungen geschlussfolgert.
-> implizite Typisierung

Beispiel:

```
var myInt = 0;
var myBool = true;
var myString = "Time, marches on ... ";
Console.WriteLine("myInt is a: {0}", myInt.GetType().Name);
Console.WriteLine("myBool is a: {0}", myBool.GetType().Name);
Console.WriteLine("myString is a: {0}", myString.GetType().Name);
```

Schleifen

wie in C# gibt es die folgenden Schleifen:

- do {...} while (...)
- while (...) {...}
- for (...; ...; ...) {...}

zusätzlich gibt es die foreach Schleife:

- foreach (Typ element in Menge)
- jedes Element einer Menge wird genau 1 mal durchlaufen
- Die Menge kann alles sein was das Ienumerable Interface implementiert (Array, Liste, ...)
- entspricht einem Iterator in C++

- durchlaufene Elemente können nicht verändert werden
- Beispiel:

```
string[] carTypes = {"Ford", "BMW", "Trabbi", "Honda" };
foreach (string c in carTypes)
  Console.WriteLine(c);
```

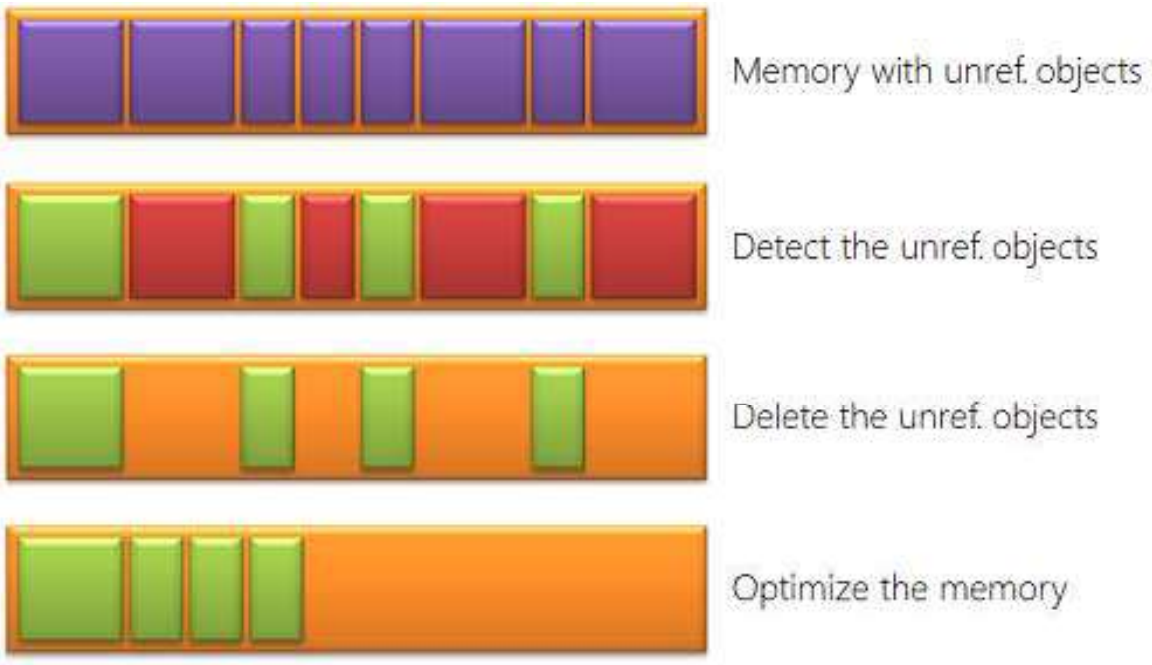
Garbage Collector

- Objekte benötigen Speicher im Heap (Speicherallokation mit new)
- werden Objekte nicht mehr benötigt sollte der Speicher wieder freigegeben werden
- Bsp:

```
Circle kreis1 = new Circle();
Circle kreis2 = new Circle();
[...]
kreis1 = null2;
```

- Speicherfreigabe ist in C# nicht explizit möglich, sondern geschieht durch den Garbage Collector

<u>Funktionsweise:</u>



Garbage Collector wird ausgelöst:

1. wenn Programm wenig ausgelastet
2. wenn Speicherressurcen knapp
3. bei Programmende
4. manuell mit GC.collect();