Window Presentation Foundation

- Dipl.-Inf., Dipl.-Ing. (FH) Michael Wilhelm
- Hochschule Harz
- FB Automatisierung und Informatik
- mwilhelm@hs-harz.de
- http://www.miwilhelm.de
- Raum 2.202
- Tel. 03943 / 659 338

Inhalt

- Einführung WPF
- Layouts
- **C# Sprache**
- Dialog-Elemente, Menüs
- 2D- / 3D-Grafik, Audio, Video, Animation
- Routet Events, Dependency Properties, Command
- Textdarstellung (Flow-FixedDocuments)
- Datenbanken
- Navigation / Browser
- Eigene Komponenten

Visual Studio, C# und .net

- Entwickler: Andreas Hejlsberg
- Heilsberg hatte Turbo Pascal und Delphi mit entwickelt
- Absprung nach Redmond
- Entwicklung des Programmpaket .net / C#
- Philosophie weitgehend identisch
  - Drag & Drop der GUI
  - Propertyfenster mit Property-Methoden kein normales get/set
  - Antwort auf Java, Plattform unabhängig
  - Sprache C#
  - Aktuelles Framework 4.5
  - Common Language Runtime (CLR)
  - Unterschied: Übersetzung, kein Intepreter,
  - Übersetzen VOR oder während der Ausführung

Visual Studio, C# und .net

- Weitere Eigenschaften
  - Unterstützt viele Programmiersprachen (C#, VB, Delphi)
  - C#, VB, Delphi, C++
  - Umwandlung in einem Zwischencode
  - Sehr viele GUI-Elemente, bis zu Listview / Grid
  - Datenbank-Anbindung
  - Web-Server
  - Delegates statt Funktionspointer
  - Operator Überladen, nicht in Java
  - Eigenschaften (readX, writeX, get/set)
  - Alles Objekte, auch int und double
  - Anweisungen weitgehend identisch zu Java / C++
  - keine Header-Dateien
  - statt implement verwendet man using
  - Mehr Datentypen, z. B. unsigned int
  - Arrays mittels Blockstruktur, anders als in Java

Visual Studio, C# und .net

- Datentypen in C#

- byte	vorzeichenlos, 0 bis 255
- sbyte	vorzeichenbehaftet, -128 bis +127
- short	vorzeichenbehaftet, -32768 bis +32767
- ushort	vorzeichenlos, 0 bis 65535
- int	vorzeichenbehaftet, -2.147.483.648 bis + 2.147.483.648
- uint	vorzeichenlos, 0 bis 4.294,967,295
- long	vorzeichenbehaftet -2^{63} bis $2^{63}-1$
- ulong	vorzeichenlos 0 bis bis $2^{64}-1$
- single	32 Bit Gleitkommazahl, 7 Stellen
- double	64 Bit Gleitkommazahl, 16 Stellen
- bool	boolscher Wert
- char	Zeichen
- decimal	96 Bit Dezimalwert
- string	readonly Zeichenfolge, neu erzeugen oder StringBuilder

Visual Studio, C# und .net

- Hello World

```
using System;
namespace bsp1
{

   public class Programm  {
     public static void Main(string[] args)    {
       Console.WriteLine("Hello 2010");
       Console.ReadLine();
     } // Programm
   }// bsp1

}
```

- Format: printf

Visual Studio, C# und .net

- Hello World

```
using System;
namespace bsp1
{
   public class Programm  {
     public static void Main(string[] args)    {
       Console.WriteLine("Hello {0} im Jahr {1}", "ix", 2010);
       Console.ReadLine();
     } // Programm
   }// bsp1

}
```

- Format: printf

Visual Studio, C# und .net

- using System // Package
- namespace // definiert einen Prefix für Definitionen
- Namespace System // int32 statt System.int32
- Console ist ein Unterbereich von System
- Console.WriteLine(…) // statt System.Console.WriteLine
- Console.ReadLine(); // wartet auf Eingabe oder Strg+F5

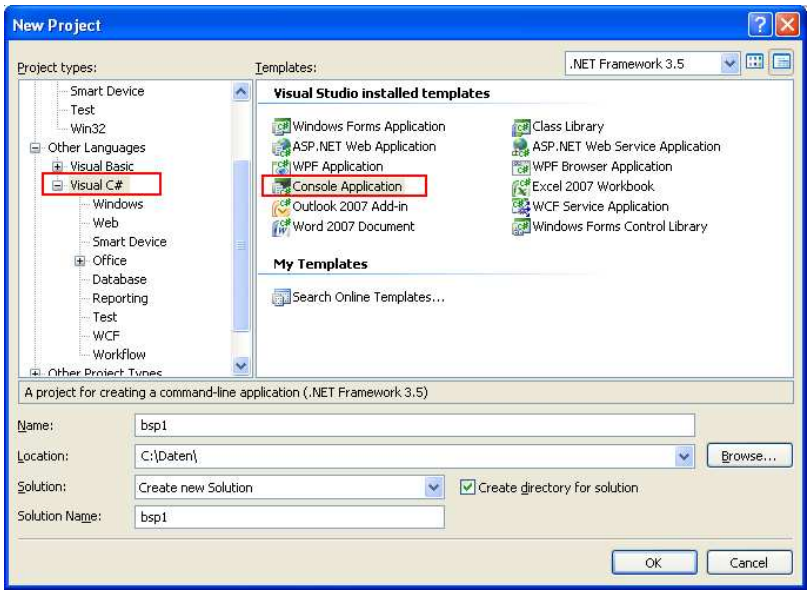
Rundungen:
double x=1234.567;
String s = x.ToString("#.##"); 1234,57
String s = x.ToString("00000"); 01235 (Rundung)
String s = x.ToString("#"); 1235 (Rundung)
String s = x.ToString("#,###"); 1.235 (Rundung, Tausender-Trennung)
String s = x.ToString("#.##E+00"); 1.23E+03 (Rundung)

Visual Studio, C# und .net

Beispiel 1

- File New Project
- Visual C#
- Console Application
- Einfügen der beiden Zeilen
  - Console.WriteLine("Hello {0} im Jahr {1}", "ix", 2010);
  - Console.ReadLine();
- Taste F5 startet das Programm oder Strg+F5

Visual Studio, C# und .net



Zeichenketten-Operationen für die Klasse string

Methode	Beschreibung
Length	Länge eines Strings
Clone	Liefert Verweis des Strings
Compare	Vergleicht zwei String (true, false)
CompareTo	Vergleicht zwei String (-1, 0, +1)
Concat	Verbindet zwei Strings
Copy	Kopieren des Strings
CopyTo	Kopieren von Zeichen
EndsWidth	Vergleich mit Ende des String (\)
Equals	Haben zwei String denselben Inhalt
Format	Printf
GetEnumerator	Funktion mit Parameter des String Durchlaufen des String

Zeichenketten-Operationen

Methode	Beschreibung
IndexOf	Sucht das erste Vorkommen
IndexOfAny	Liefert alle Vorkommen in einer Liste
Insert	Zeichen einfügen
Join	Die Methode fügt zwischen je zwei Elementen eines Strings-Arrays einen angegebenen trennenden Strings ein und liefert das Ergebnis a,b,c ⇒ a_b_c
LastIndexOf	Sucht das letzte Vorkommen
LastIndexOfAny	Liefert alle Vorkommen in einer Liste Umgekehrte Reihenfolge
PadLeft	Rechtsbündige Ausrichtung, Blanks links
PadRight	Linksbündige Ausrichtung, Blanks rechts
Remove	Löschen von Zeichen
Replace	Ersetzen von Zeichen

Zeichenketten-Operationen

Methode	Beschreibung
Split	Aufteilen (Parsen oder \)
StartsWith	Prüft die ersten Zeichen auf Übereinstimmung
Substring	Liefert einen Teilstring
ToCharArray	Umwandlung von String in Char-Array
ToLower	Kleinbuchstaben
ToUpper	Großbuchstaben
Trim	Löschen von Leerzeichen am Anfang und am Ende
TrimEnd	Löschen von Leerzeichen am Ende
TrimStart	Löschen von Leerzeichen am Anfang
Convert	Convert.ToInt32(sStr); Convert.ToBoolean Convert.ToDateTime Convert.ToDecimal Convert.ToDouble

Beispiel: String 2 Double

```
string[] values= { "-1,035.77219", "-1035.77219", "-1035,77219",
                   "1AFF",                        "1e-35",
                   "1,635,592,999,999,999,999,999,999",
                   "-17.455",                     "190.34001",
                   "1.29e325"};
double result;
foreach (string value in values) {
   try {
         result = Convert.ToDouble(value);
         Console.WriteLine("Converted '{0}' to {1}.", value, result);
   } catch (FormatException) {
         Console.WriteLine("Konvertierungsfehler '{0}' to a Double.", value);
   }
   catch (OverflowException) {
         Console.WriteLine("
                   '{0}' ist außerhalb des Bereiches eines Doubles.", value);
   }
}
```

Ergebnis: String 2 Double

```
C:\WINDOWS\system32\cmd.exe
Konvertierungsfehler '-1,035.77219' to a Double.
Converted '-1035.77219' to -103577219.
Converted '-1035,77219' to -1035,77219.
Konvertierungsfehler '1AFF' to a Double.
Konvertierungsfehler '0x1AFF' to a Double.
Converted '1e-35' to 1E-35.
Konvertierungsfehler '1,635,592,999,999,999,999,999,999' to a Double.
Converted '-17.455' to -17455.
Converted '190.34001' to 19034001.
'1.29e325' ist außerhalb des Bereiches eines Doubles.
```

- -1035.77219 wird als kommalose Zahl interpretiert
- -1035,77219 wird als Komma-Zahl interpretiert

- sStr = "-1234.4567";
- x = Convert.ToDouble(sStr, System.Globalization.CultureInfo.InvariantCulture);

Ergebnis2: String 2 Double

```
C:\WINDOWS\system32\cmd.exe
Converted '-1,035.77219' to -1035,77219.
Converted '-1035.77219' to -1035,77219.
Converted '-1035,77219' to -103577219.
Konvertierungsfehler '1AFF' to a Double.
Konvertierungsfehler '0x1AFF' to a Double.
Converted '1e-35' to 1E-35.
Converted '1,635,592,999,999,999,999,999,999' to 1,635593E+24.
Converted '-17.455' to -17,455.
Converted '190.34001' to 190,34001.
'1.29e325' ist außerhalb des Bereiches eines Doubles.
```

string hexOutput = String.Format("{0:X}", value);

Plausibilitätsprüfung

```
sStr=ENote.Text.trim();
double note;
bool bcanConvert = double.TryParse(sStr, out note);
if ( ! bcanConvert)
{
   ENote.Focus();
   MessageBox.Show("Fehlerhafte Eingabe, die Note ist keine Zahl"
     , "Note", MessageBoxButtons.OK);
     return;
}

if (double.TryParse(textBox1.Text, NumberStyles.AllowDecimalPoint,
      System.Globalization.CultureInfo.InvariantCulture, out value1) ) {
}
```


C# Sprache

- Formatierte Ausgabe:

```
double d = 1.0 / 3.0;
// #  leer oder zahl
// 0  0 oder zahl  besser ###0.00 slas ###.##
Console.WriteLine("d: {0:##0.00}", d);
```

Format-Code	Beschreibung
{0} {1} {2}	Zahl mit mehreren Nachkommastellen
{0:##0.00}	Zahl mit zwei Nachkommastellen Dezimaltrennzeichen ist ein Punkt
{0:p2}	Zahl mit zwei Nachkommastellen Dezimaltrennzeichen ist ein Komma
{0,7:c}	Ausrichtung
("").PadRight(24, '-')	Ergibt 24 waagerechte Striche

Formatierungen

double x=1234.567;

Edit.Text = x.ToString("#.##"); 1234,57
Edit.Text = x.ToString("00000"); 01235 (Rundung)
Edit.Text = x.ToString("#"); 1235 (Rundung)
Edit.Text = x.ToString("#,###"); 1.235 (Rundung,Tausender-Trennung)

Edit.Text = x.ToString("#.##E+00"); 1.23E+03 (Rundung)

Path-Character:

string sPath = @"c:\daten\test.jpg";

string sPath = "c:\\daten\\test.jpg";

C# Sprache

Weitgehend identisch mit Java,

- If-else-Anweisung identisch
- Switch-case-Anweisung unterschiedlich
- Liefert eine Fehlermeldung, break fehlt

```
switch (i)  {
        case 0: erg = a + b;
        case 1: erg = a - b;
        case 2: erg = a * b;

    }
```

- Nun okay, mit leerer Anweisung

```
switch (i)  {
        case 0:
        case 1: erg = a - b;        // korrekt ohne Anweisung
                break;
        case 2: erg = a * b;
                break;
    }
```

C# Sprache

- For-Schleife — identisch
- foreach-Schleifen und Iteration — neu, aber ähnlich in Java

```
int [] array = { 1,2,3,4,5 };
foreach (int i in array) {
   Console.WriteLine(i);
}
```

- While-Schleife — identisch
- Do-While-Schleife — identisch


```
// Eindimensionales Array
      int[] numbers = new int[5];

// Mehrdimensionales Array
      string[,] names = new string[5,4];

// Array von Arrays (verzweigtes Array, à la Java)
      byte[][] scores = new byte[5][];

 // Verzweigtes Array erstellen.
      for (int i = 0; i < scores.Length; i++)
      {
         scores[i] = new byte[i+3];
      }

// Länge der einzelnen Zeilen ausgeben.
      for (int i = 0; i < scores.Length; i++)
      {
         Console.WriteLine("Length of row {0} is {1}", i, scores[i].Length);
      }
```

C# Sprache

- Enumerationen neu
- Automatische Konstanten mit ganzzahligem Wert

```
enum FB {
    AI,
    VW,
    W
}
enum FB {
    AI=0x0000FF,
    VW=0xFF0000,
    W=0x00FF00
}
```

```
FB f;
f = FB.AI;
int i = (int)f;
i = (int)FB.W;
```

```
static void testfor() {
  foreach (int j  in Enum.GetValues(typeof(FB)))  {
     Console.WriteLine("Fachbereich {0} {1}",
       Enum.GetName(typeof(FB),j),j);
  }
}  // Ausgabe ???
```

Enum mit Zahlen: serialize

- FB.IsDefined(typeof(FB), i)

C# Sprache

- Exception fast identisch
- public double calc(double i) {

```
    try {
        double j = sqrt(i);
    } catch (MyException me)
    {
        Console.WriteLine(me.m_msg);
    } catch {
        /* alle restlichen  Exceptions */
    } finally {
        /* Aufraeumen */
    }
} // calc
```

C# Sprache: Datum und Zeit

- DateTime dt1 = new DateTime(2010, 04, 28);
- Console.WriteLine(dt1); // 2String: 28.04.2010 00:00:00

- DateTime dt2 = new DateTime(2010, 04, 28, 22, 45, 12);
- Console.WriteLine(dt2); // 2String:

- DateTime dt3 = new DateTime(2010, 04, 28, 22, 45, 12,99);
- // Millisekunden

- Aktuelle Datum:
- DateTime dt4 = DateTime.Now
- Console.WriteLine(dt4);
- Console.WriteLine("Datum: {0} Tag:{1} Monat:{2}
  Jahr:{3} ", dt4, dt4.Day, dt4.Month, dt4.Year);

datetime1

C# Sprache: Performance

```
static private void test5()
{
    double dValue;
    DateTime Time1, Time2;
    Time1 = DateTime.Now;
    for (int i=0; i<10000; i++)
        for (int j = 0; j < 10000; j++)
            dValue = Math.Sin(12.345);
    Time2 = DateTime.Now;
    TimeSpan duration1 = Time2 - Time1;
    Console.WriteLine(duration1);  // 4,3 Sekunden
}
```

Mathematische Funktionen: Math.Sin

Name	Description
Abs	Returns the absolute value of a specified number.
Acos	Returns the angle whose cosine is the specified number.
Asin	Returns the angle whose sine is the specified number.
Atan	Returns the angle whose tangent is the specified number.
Atan2	Returns the angle whose tangent is the quotient of two specified numbers.
BigMul	Produces the full product of two 32-bit numbers.
Ceiling	Returns the smallest integer greater than or equal to the specified number.
Cos	Returns the cosine of the specified angle.
Cosh	Returns the hyperbolic cosine of the specified angle.
DivRem	Calculates the quotient of two numbers and also returns the remainder in an output parameter.
Exp	Returns e raised to the specified power.
Floor	Overloaded. Returns the largest integer less than or equal to the specified number.
IEEERemainder	Returns the remainder resulting from the division of a specified number by another specified number.
Log	Returns the logarithm of a specified number.
Log10	Returns the base 10 logarithm of a specified number.
Max	Returns the larger of two specified numbers.
Min	Returns the smaller of two numbers.
Pow	Returns a specified number raised to the specified power.
Round	Rounds a value to the nearest integer or specified number of decimal places.
Sign	Returns a value indicating the sign of a number.
Sin	Returns the sine of the specified angle.
Sinh	Returns the hyperbolic sine of the specified angle.
Sqrt	Returns the square root of a specified number.

C# Sprache

- Parameter-Übergabe an eine Methode: 4 Typen
- a) Parameter per Value
- b) Parameter per Referenz (ref)
  Kann sofort gelesen oder geschrieben werden
- c) Parameter für die Ausgabe (out)
  Muss erst gesetzt werden, dann darf gelesen werden
- d) Parameter für Arrays (params)
  muss immer am Ende der Liste stehen
  nur ein einfaches Feld, keine Matrix
  auch mit drei Zahlen möglich

C# Sprache

```
static void add2(int a, int b, out int erg)    {
   int d=erg;
    erg = a + b;
   d=erg;
}
static void add3(int a, ref int b)   {
   b = a + b;
}int a = 11;
 int b = 22;
 int c;                    c=add1(a,b);
 add2(a, b,out c);      // out signalisiert Ausgabeparameter
 Console.WriteLine("a {0}   b {1}  c{2}", a, b, c);
 a = 11;
 b = 22;
 add3(a, ref b);         // ref signalisiert Referenzparameter
 Console.WriteLine("a {0}   b {1}", a, b);
```

bsp3

C# Sprache: Parameter mit params

```
void ShowNumbers (params int[] numbers) {
foreach (int x in numbers) {
        Console.Write (x+" ");
}
Console.WriteLine();
}

int[] x = {1, 2, 3};
ShowNumbers (x);
ShowNumbers (4, 5);
ShowNumbers (4, 5,6,7,8);
```

C# Sprache: struct

- Vorteile
  - Zusammenfassen von Daten
  - Keine Klasse
  - Als Parameter für Methoden geeignet

```
public struct Student
{
    public string name;
    public int matrnr;
    public int unummer;
}
```


C# Sprache: struct

```
static void Main(string[] args)
{
    Student s1, s2;
    s1.name = "Bender";
    s1.matrnr = 1234;
    s1.unummer = 23233;
    s2 = s1;  // kopieren
    Console.WriteLine("Studentendaten");
    Console.WriteLine("Name: {0}  Matrnr: {1}  UNummer: U{2}",
        s2.name, s2.matrnr, s2.unummer);
    Console.ReadLine();
}
```

C# Sprache: struct

```
static void print(Student std)
{
        Console.WriteLine("Studentendaten");
        Console.WriteLine("Name: {0}  Matrnr: {1}  UNummer: U{2}",
                std.name, std.matrnr, std.unummer);
         Console.ReadLine();
}
```

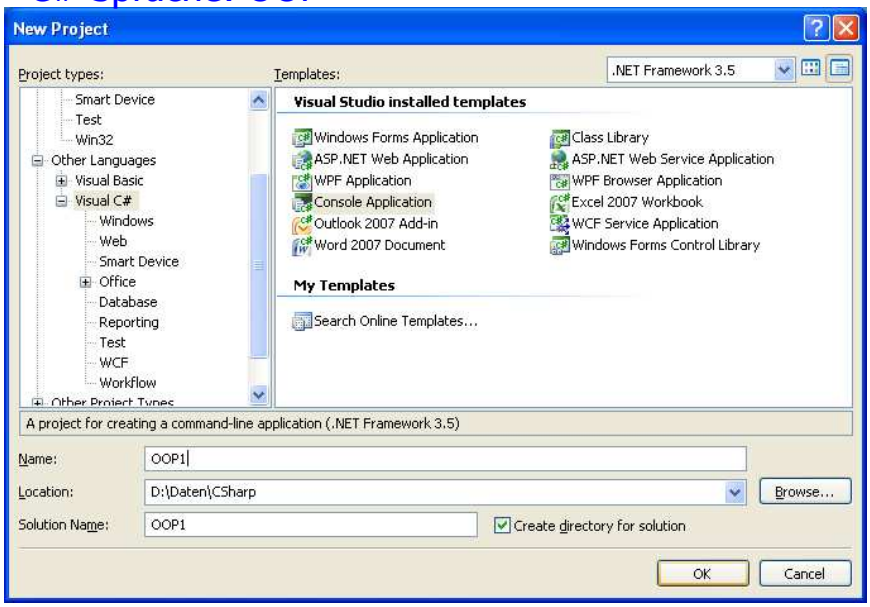

C# Sprache: OOP

- Zugrif auf Variablen:

```
private
internal  // nur in Assembly, Packages
public
protected
```

Modifizierer	Innerhalb der Klasse	Innerhalb der Ableitungen	Außerhalb der Klasse	Außerhalb der Anwendung
public	Ja	Ja	Ja	Ja
internal	Ja	Ja	Ja	Nein
protected	Ja	Ja	Nein	Nein
private	ja	Nein	Nein	Nein

C# Sprache: OOP



C# Sprache: OOP

- Konstruktor wie java
- Dekonstruktor fast wie java

```
~student()
{
}
```

- Methoden: set/get-explizit
- Ableitung einer neuen Klasse:

```
class linie  : punkt
{
   public linie(int x, int y): base(y,x)
  {
   }
}
```

C# Sprache: OOP

```
class student   {
    private int m_Matrnr;

    public student()
    {
      m_Matrnr = 0;
    }
    public student(int Matrnr)
    {
      m_Matrnr = Matrnr;
    }
    ~student()
    {
      m_Matrnr = 0;
    }
```

```
class student   {
    private int m_Matrnr;

    public student():this(0)
    {
    }
    public student(int Matrnr)
    {
      m_Matrnr = Matrnr;
    }
    ~student()
    {
      m_Matrnr = 0;
    }
```


C# Sprache: OOP

```
class student   {
    private int m_Matrnr;
    public int Matrnr
    {
      get
      {
        return m_Matrnr;
      }
      set
      {
        if ( value>0 && value < 99999 ) m_Matrnr=value;
      }
    }
    public void print()
    {
      Console.WriteLine(m_Matrnr);
    }
  }
```

```
public void test
{
  int k;
  student  std = new student();
  std.Matrnr=17421;  // ruft set-Methode auf
  k = std.Matrnr;  // ruft die get-Methode auf
}
```

C# Sprache: OOP

```
class Program
  {
    static void Main(string[] args)
    {
      student hase, igel;  // keine Erzeugung, kein C++
      hase = new student();
      hase.Matrnr = 14721;
      hase.print();
      igel = new student(12345);
      igel.print();
      Console.ReadLine();
    }
  }
```


C# Sprache: Mehrere Konstruktoren (this/base)

```
class koerper   {
    private int m_volume;

    public koerper():this(0)      {
    }

    public koerper(int volume)      {
      m_volume = volume;
    }

    public string getVolumen()      {
        return m_volumen;
    }
}
```

C# Sprache: Überscheiben von Methoden

```
class koerper    {
      private string m_volume;
      public koerper(int volume)        {
         m_name = name;
      }
      public string name        {
         get          {
            return m_name;
         }
      }
      virtual public void print()        {
         Console.WriteLine("Name: {0}  Volumen {1}", m_name, volume);
      }
   }
```


C# Sprache: Überschreiben von Methoden

```
class koerper2 : koerper    {
      private string m_name;
      public koerper2(string name, int volume):base(volume)        {
         m_name = name;
      }
      public string name        {
         get          {
            return m_name;
         }
      }
      public override void print()        {
         Console.WriteLine("Name: {0}  Volumen {1}", m_name, volume);
      }
   }
```

public virtual void print()

C# Sprache: Operator Überladen

Operatoren	Overloadbar
+, -, *, /, %, &, |, <<, >>	Alle binären Operatoren können überladen werden
+, -, !, ~, ++, --, true, false	Alle unären Operatoren können überladen werden
==, !=, <, >, <= , >=	Alle relationalen Operatoren können, paarweise, überladen werden
&&, ||	Keine Überladung
()	Cast-Operator
+=, -=, *=, /=, %=	Können überladen werden. Funktioniert aber automatisch
=, . , ?:, ->, new, is, as, sizeof	Keine Überladung

C# Sprache: Operator Überladen

```
class CKoerper
{
    private int m_volume;

    public static CKoerper operator +(CKoerper k1, CKoerper k2)
    {
      CKoerper temp = new CKoerper();
      temp.volume = k1.volume + k2.volume;
      return temp;
    }

}
```

C# Sprache: is statt instanceOf

```
CQuadrat q1 = new CQuadrat(5);
CRechteck r1 = new CRechteck(5, 3);
CKreis k1 = new CKreis(5);
ArrayList arr = new ArrayList();
arr.Add(q1);            arr.Add(r1);            arr.Add(k1);


foreach (Object obj in arr)   {
   if (obj is CQuadrat)
   {
      CQuadrat q = (CQuadrat)obj;
   }
   if (obj is CRechteck)
   {
      CRechteck r = (CRechteck)obj;
   }
}
```


C# Sprache: Allgemeine Container

- Array // feste Anzahl
- ArrayList // System.Collections
  - Methoden: Count, Capacity
- ListDictionary // System.Collections.Specialized
  Schlüssel/Werte in einer Liste
- Hashtable
- Stack
- Queue
- BitArray

C# Sprache: Generische dynamische Container

- List<T>
- SortedList<T>
- Dictionary<T> // System.Collections.Specialized Schlüssel/Werte in einer Liste
- Stack<T> //
- Queue<T> //

- List<CStudent> liste = new List<CStudent>();
- liste.Add(new CStudent("Meier");
- liste.Add(new CStudent("Schmidt");
- Foreach (cStudent std in liste)
- Console.WriteLine(std+"");


```
using System.Collections;
public class BeispielArrayList {
public static void Main() {
      // Erzeugen und Init des ArrayList.
      ArrayList myliste = new ArrayList();
      myliste.Add("Hello");
      myliste.Add("World");
      myliste.Add("!");
      // Anzeige
      Console.WriteLine( "myListe" );
      Console.WriteLine( " Count: {0}", myliste.Count );
      Console.WriteLine( " Capacity: {0}", myliste.Capacity );
      Console.Write( " Values:" );
      PrintValues( myliste );
}
public static void PrintValues( IEnumerable myList ) {
    foreach ( Object obj in myList )
       Console.Write( " {0}", obj );
       Console.WriteLine();
  }
}
```

C# Sprache: Dynamische Container

- Hashtable htliste = new Hashtable();
- CStudent s1, s2, s3, s4;
- s1 = new CStudent(…);
- htliste[s1.Matrnr] = s1;
- htliste[s2.Matrnr] = s2;
- htliste.Add(s3.Matrnr , s3);

- s4 = (CStudent) htliste[14721]; // holen des Studenten

C# Sprache: Delegates

- Delegates sind der Ersatz für Pointerfunktionen

- Delegates können sowohl statische als auch Instanzmethoden referenzieren.

- Sie müssen aber denselben Rückgabewert und dieselben Parametertypen besitzen wie die zu referenzierenden Methoden.

```
public delegate int calcDel(int val1, int val2);  // "Deklaration der Funktion"

public class Programm    {
   public static int Addint(int a, int b)
   {
      return a + b;
   }

   public static int Multint(int a, int b)
   {
      return a * b;
   }

 calcDel calc1  = new calcDel(Addint);
     calcDel calc2  = new calcDel(Multint);
      test1(calc1);
      test1(calc2);

public static void test1(calcDel calc)
{
    int[] a = {1,2,3,4,5};
    int i;
    for (i=0; i<a.Length; i++)
    {
       a[i] = calc(a[i], 2);
    }
    i=0;
    foreach (int j in a)
    {
      i++;
      Console.WriteLine("A[ {0} ] = {1}", i, j);
    }
    Console.WriteLine();
}
```



```
public static void Main()
{
      // Die Instanzmethode dem Delegate zuweisen
      calcDel calc1  = new calcDel(Addint);
      calcDel calc2  = new calcDel(Multint);
       test1(calc1);
       test1(calc2);

     Console.ReadLine();
}


}  // Programm
```

wpfwpf

Delegates in Visual Basic

```
'-------------------------------------Delegated
  Delegate Sub SetProgressbarCallback(ByVal i As Integer) 'Rueckrufmethode

  Public Sub SetProgressbar1(ByVal i As Integer) 'Werte setzen
    If Me.ProgressBar1.InvokeRequired Then 'wenn schon in Benutzung
          'Delegate anlegen
      Dim d As New SetProgressbarCallback(AddressOf SetProgressbar1)
      Me.Invoke(d, i) 'Delegate senden
    Else
      Me.ProgressBar1.Value = i ' wenn nicht benutzt -> Wert zuweisen
    End If
  End Sub
'-------------------------------------#Delegated
```


C# Sprache: **Literatur und Links**

Softwareentwicklung mit C#
Hanspeter Mössenböck
dpunkt.Verlag
ISBN 3-89864-406-5

Visual C+ 2005
Günter Born, Benjamin Born
Entwickler.press
ISBN 978-3-939084-40-2

- http://www.guidetocsharp.de
- http://msdn.microsoft.com/de-de/library/kx37x362.aspx