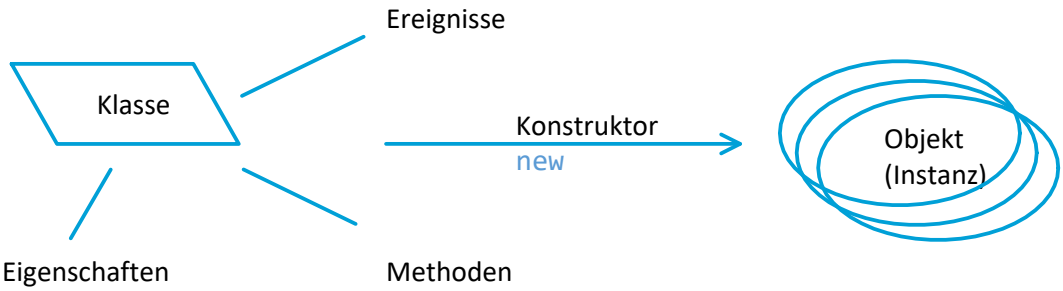
Objekt-Orientierte Programmierung

Mittwoch, 11. September 2024 08:51




```
1  // Konstruktor-Aufruf
2  Radio radio = new Radio();
3
4  // Methode
5  radio.Einschalten();
6  radio.SenderWechseln("WDR 4");
7  radio.SenderWechseln(sender: "Radio BOB", carrier: 1);
8  byte[] stream = radio.GetAudioStream(url: "http://....");
9
10 // Eigenschaften (Properties)
11 radio.Sender = "1Live";
12 string sender = radio.Sender;
13
14
15
16 public class Radio
17 {
18         public Radio()
19         {
20                 // Konstruktor
21         }
22
23         public void SenderWechseln(int carrier, string sender)
24         {
25                 // Code zum Senderwechseln
26         }
27
28         public byte[] GetAudioStream(string url)
29         {
30                 // Code zum Streamen
31         }
32
33         // Backing Field für die Sender-Property
34         private string _sender
35         // Öffentliche Eigenschaft (Property)
36         public string Sender
37         {
38                 get // wenn Wert abgefragt wird
39                 {
40                         return _sender;
41                 }
42                 set // wenn Wert zugewiesen wird
43                 {
44                         _sender = value;
45                 }
46         }
47
48         public single Frequenz;
49
50         public int Lautstaerke { get; set; }
}
```

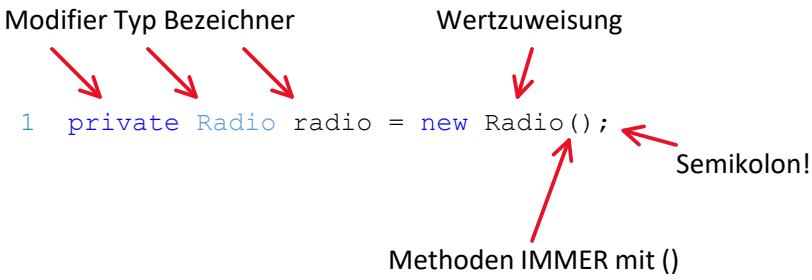
Full qualified property
(Snippet: propfull)

Öffentliches Feld

Auto-Property

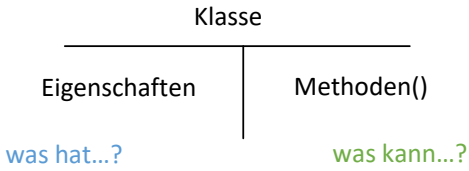
C# - Grundlagen

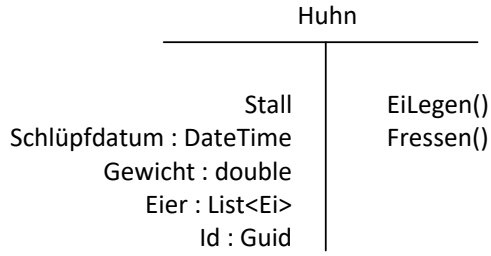
Mittwoch, 11. September 2024 11:32

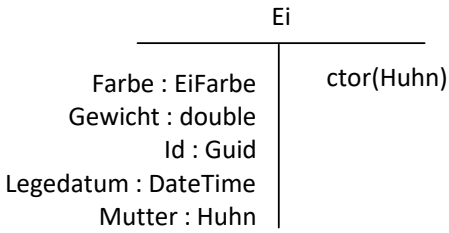


Objektmodell Eierfarm

Mittwoch, 11. September 2024 11:51







Array

Mittwoch, 11. September 2024 13:14

```
string feld[] = new string[] { "Blau", "Weiß", "Rot" };
```

Feld[]

Index	Feld
0	"Blau"
1	"Weiß"
2	"Rot"

```
string zwei = feld[1];

for (int i=0; i < feld.Length;i++)
{
      Console.WriteLine(feld[i]);
}

feld[4] = "Grün"; // IndexOutOfRangeException!
```

Datentypen

Mittwoch, 11. September 2024 13:31

Zahlen

Ganzzahl	short, int, long int16, int32, int64 uint16, uint32, uint64
Fließkommazahl	float single, **double**, decimal
Wahrheitswert	bool

*alle "primitiven" Datentypen sind mit 0 vorinitialisiert

Zeichen

String	Zeichenkette
Char	Zeichen

*Strings in C# sind mit null vorinitialisiert

Nullable in C#

Mittwoch, 11. September 2024 15:30

Skalartypen

```
// Nicht möglich:
int a1 = null;

// Lösung:
System.Nullable<int> a2 = null;
int? a3 = null;

// Problem jetzt:
int b1 = 12;
int c1 = b1 + a3; // knallt, weil Typen unverträglich!
// stattdessen:
int c2 = b1 + (a3.HasValue ? a3.Value : -99);
int c3 = b1 + a3 ?? -99;
```

Referenz-Typen (komplexe Typen)

```
Huhn huhn1;

huhn1.Name = "Hilde"; // Compilerwarnung: huhn1 könnte hier null sein

if (huhn1 != null) // Null-Prüfung
{
      huhn1.Name = "Hilde"; // Keine Warnung, weil Null-Prüfung vorweg
}

Huhn? huhn2; // Nullable-Huhn, kann null enthalten!

huhn2.Name="Hilde"; // Compilerwarnung: huhn1 könnte hier null sein

// Inline-Null-Prüfung
int? ergebnis1 = (huhn2 != null ? huhn2.Eilegen() : null);
int? ergebnis2 = huhn2?.Eilegen();
```

Konstruktoren in C#

Mittwoch, 11. September 2024 16:00

- enthält eine Klasse keinen Konstruktor → Parameterloser Standardkonstruktor vorhanden
- enthält eine Klasse einen/mehrere explizit programmierten Konstruktor(en) → nur der/die explizit vorh. Konstruktoren existieren.

Fall-Unterscheidungen in C#

Donnerstag, 12. September 2024 09:18

```
a = (a > 12 ? a+1 : a-1); // Inline-If

if (a > 12 && b=10) return; // Einzeiliges If

if (a > 12) // If-Blöcke
{
        // Anweisungen, wenn Bedingung true
}
// optional
else if !(a < 0)
{
        // Anweisung, wenn Bedingung true
}
// optional
else
{

}
```

```
switch (this.Farbe)
{
    case EiFarbe.Hell:
      // Anweisungen
        break;
    case EiFarbe.Dunkel:
      // Anweisungen
        break;
    case EiFarbe.Gruen:
      // Anweisungen
        break;
    default:
      // Anweisungen
        break;
}
```

Vergleichsoperatoren in C#

Größer als	>
Größer oder gleich	>=
Kleiner als	<
Kleiner oder gleich	<=
Gleich	==
Ungleich	!=

Logische Operatoren

AND (UND)	&&
OR (ODER)	||
XOR (entweder oder)	|
AND ALSO	&
NOT	!

eXtensible Markup Language

Donnerstag, 12. September 2024 09:52

Character-separated values (CSV)

```
Name;Strasse;Ort;Alter
Tim;Weg 1;Wetter;9
Tom;Parallelstr. 1;Bochum;12


24-11-09
USD;1.1043
JPY;123,45

24-11-08
…
```

XML

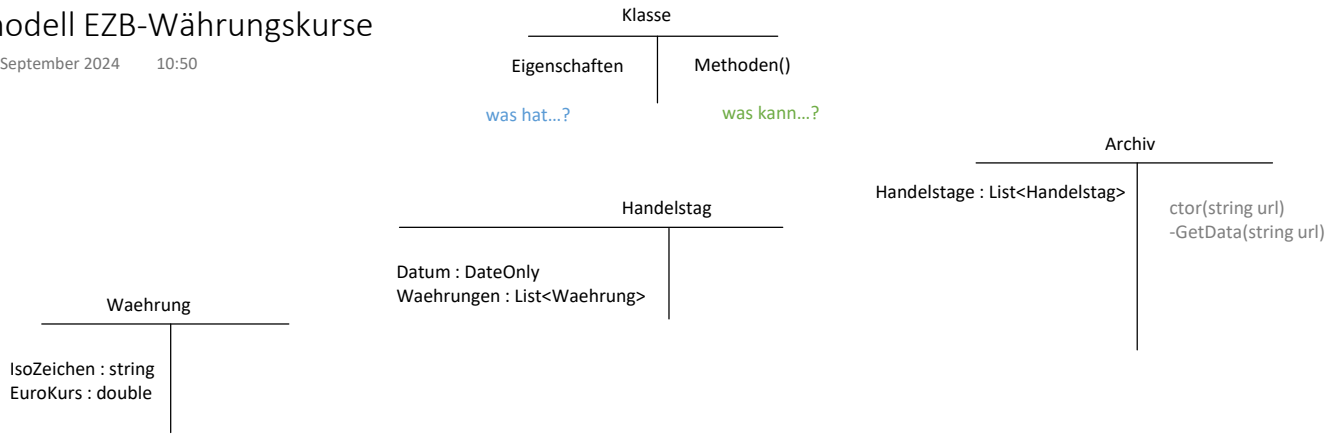
```xml
<!-- XML besteht aus Tags und Content -->
<Tag>Content</Tag>

<!-- Content kann aus Tags bestehen -->
<Tag1>
        <Tag2>Content</Tag2>
</Tag1>

<!-- Tags ohne Content (singulärer Tag) -->
<Tag />

<!-- Tags können Attribute haben -->
<Tag Attribut1="Wert1" Attribut2="Wert2">Content</Tag>
<Tag Attribut1="Wert1" Attribut2="Wert2" />

<!-- Beispiel -->
<Personen>
        <Person>
                <Name>Tim</Name>
                <Strasse>Weg 1</Strasse>
                <Alter>9</Alter>
        </Person>
        <Person>
                <Name>Tom</Name>
                <Strasse>Parallelstr.  1</Strasse>
                <Alter>12</Alter>
        </Person>
</Personen>
```

XML in C#: XDocument

```
XDocument doc = XDocument.Load("<Pfad>");

// Methoden für XElement
var q = doc.Root.Descendants() // alle(!) Unterknoten in Dokumentreihenfolge
var q = doc.Root.Elements() // alle Unterknoten der ersten Ebene

var countries = doc.Root.Descendants("country");
var pop = doc.Root.Elements("population");

// Beispiel
var data = mondialDoc.Root.Descendants("country")
                    .Select(co => new
                    {
                        Name = co.Elements("name").First().Value,
                        Bevoelkerung = co.Elements("population").Last().Value
                    });

foreach (var item in data)
{
    Console.WriteLine(item.Name + ": " + item.Bevoelkerung);
}
```

Objektmodell EZB-Währungskurse

Donnerstag, 12. September 2024 10:50



Nützliche Tastenkombinationen

Donnerstag, 12. September 2024 11:45

Codebearbeitung

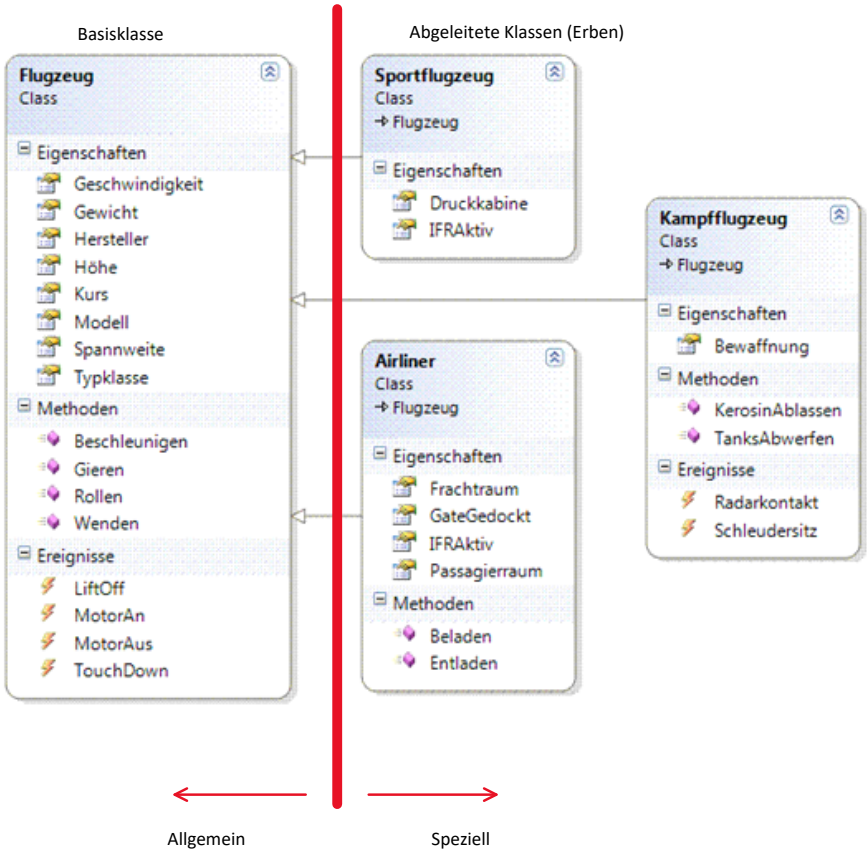
SmartTag-Menü öffnen	Strg+ . (Alt+Enter)
Zu Definition gehen	F12
Zurück	Strg+ -
Zeilen auskommentieren	Strg+K Strg+C
Auskommentierung aufheben	Strg+K Strg+U
Dokument formartieren	Strg+K Strg+D

Debuggen

Debugging starten	F5
Haltepunkt setzen	F9
Schrittweise	F11
Prozedurschritt	F10
Rücksprung	Shift+F11

Vererbung

Donnerstag, 12. September 2024 13:57



- Der Erbe kann/hat alles, was die Basisklasse auch kann/hat (außer Konstruktoren).
- Geerbt wird alles (kein Rosinenpicken!).
- Basisklasse, die von weiterer Basisklasse erbt, ist möglich.
- Erben von mehreren Basisklassen ist **nicht** möglich (→ Interface)

Language INtegrated Query

Donnerstag, 12. September 2024 14:57

SQL

```
1  select * from Customers where City like 'Witten%'
```

C# mit LINQ

```
List<Customer> customers = GetAllCustomers();

// Deklarative Syntax
var q1 = from cu in customers
                        where cu.City.StartsWith("Witten")
                        select cu;

// Methoden-/Lambda-Syntax
var q2 = customers.Where(cu => cu.City.StartsWith("Witten"));

// Zugriff auf die Ergebnismenge
foreach (var element in q1)
{
      ...
}

List<Customer> found = q2.ToList();
```

Typdeklaration durch Wertzuweisung

IEnumerable

IEnumerable (IQueryable)

Deferred Execution

Zugriff auf die Ergebnismenge
führt GetEnumerator von IEnumerable aus

C# ohne LINQ

```
List<Customer> customers = GetAllCustomers();

List<Customer> wittener = GetCustomersByCity("Witten");



public List<Customer> GetCustomersByCity(string city)
{
        List<Customer> found = new List<Customer>();

        foreach (Customer cu in customers)
        {
                if (cu.City.StartsWith(city))
                {
                        found.Add(cu);
                }
        }
        return found;
}
```

"Strings are immutable!"

Freitag, 13. September 2024 11:04

```
string hallo = "Hallo";
string welt = "Welt!";

// Böse, weil string = char[]!
string meldung1 = hallo + " " + welt;

// Seit .NET 2.0
StringBuilder builder = new StringBuilder(hallo);
builder.Append(" ");
builder.Append(welt);
string meldung2 = builder.ToString();
// oder
string meldung3 = String.Format("{0} {1}", hallo, welt);

double zahl = 12345.678;
DateTime jetzt = DateTime.Now;
string formatiert = String.Format("{0:#,##0.00} - {1:dd.MM.yyyy HH:mm}",
zahl, jetzt);

// Seit .NET 4.5
string meldung4 = $"{hallo} {welt}";
string formatiert2 = $"{zahl:#,##0.00} - {jetzt:dd.MM.yy}";
```