Zusammenfassung .NET

Inhalt

[.NET-Architektur 1](#_Toc473043495)

[Assemblies 2](#_Toc473043496)

[Common Type System 3](#_Toc473043497)

[C# Common 3](#_Toc473043498)

[GC, Destruktion & IDisposable 9](#_Toc473043499)

[Vererbung 13](#_Toc473043500)

[Delegates & Events 16](#_Toc473043501)

[Generics 18](#_Toc473043502)

[Iteratoren 20](#_Toc473043503)

[Extension Methods 22](#_Toc473043504)

[Exceptions 23](#_Toc473043505)

[LINQ 24](#_Toc473043506)

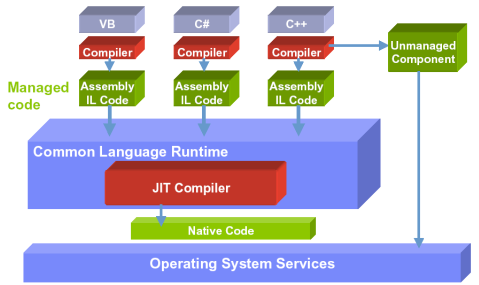
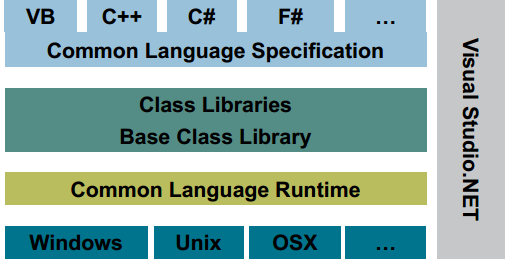
[Entity Framework 29](#_Toc473043507)

[WCF 35](#_Toc473043508)

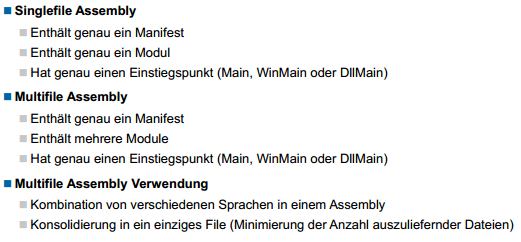
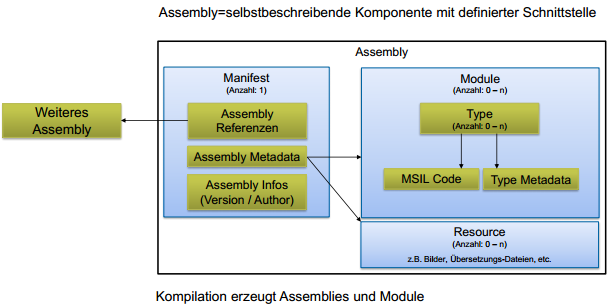
[Reflection 41](#_Toc473043509)

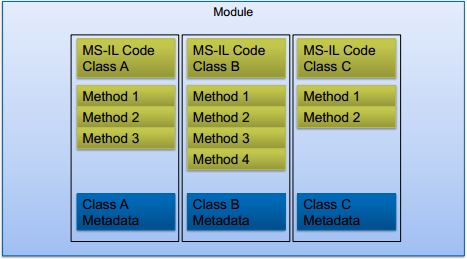
[Attributes 42](#_Toc473043510)

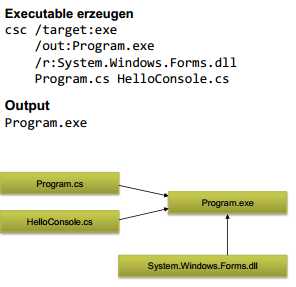
# .NET-Architektur



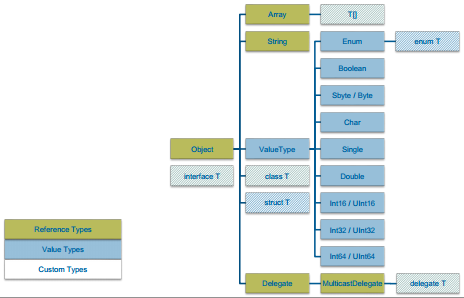
# Assemblies

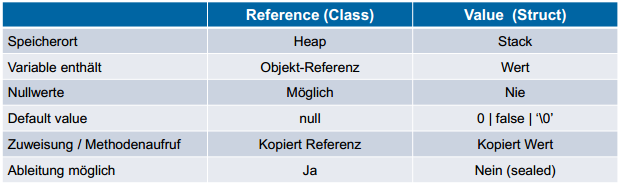




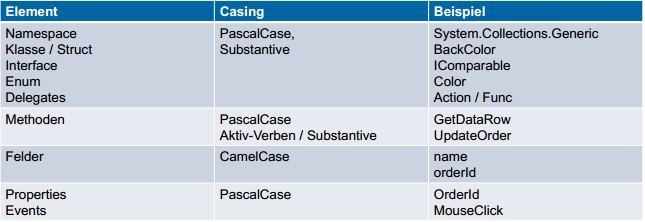


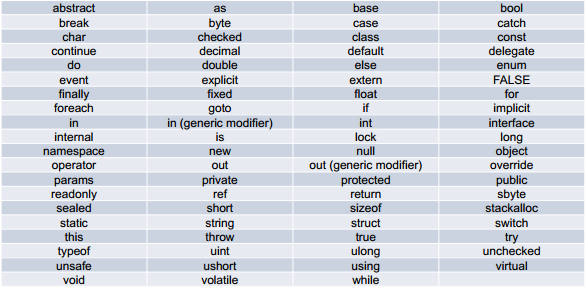
# Common Type System

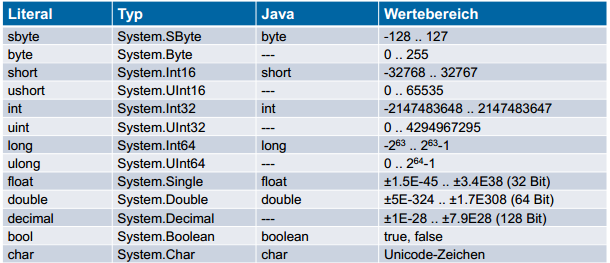


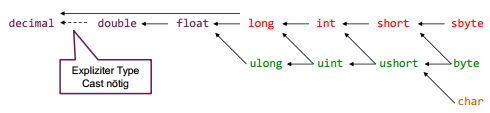


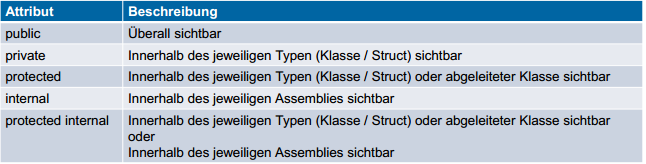
# C# Common

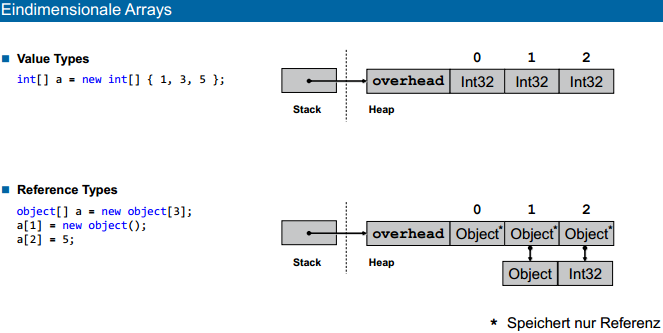


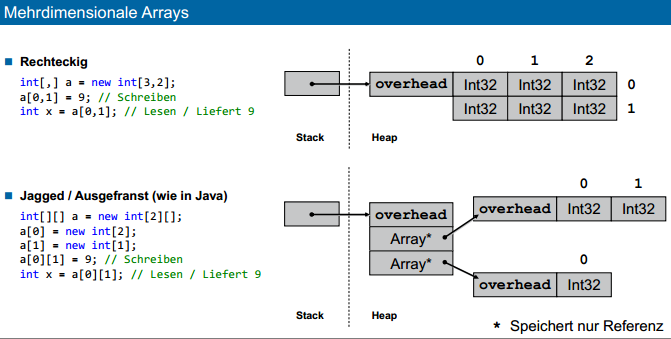


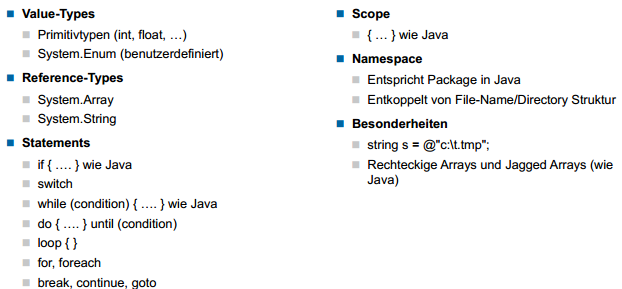


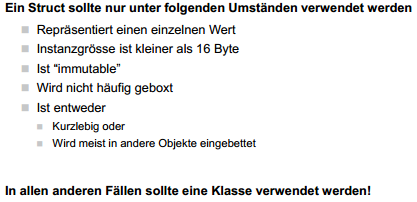


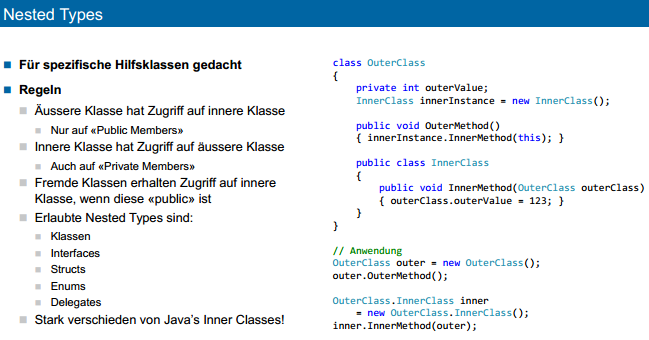






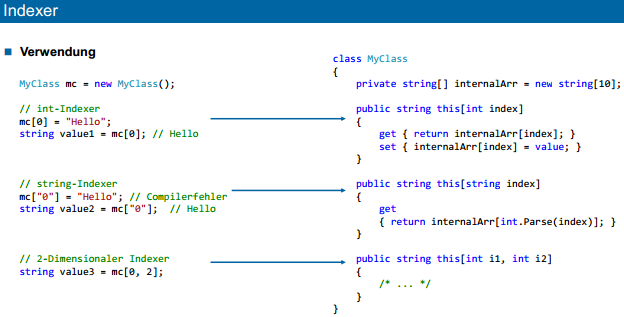




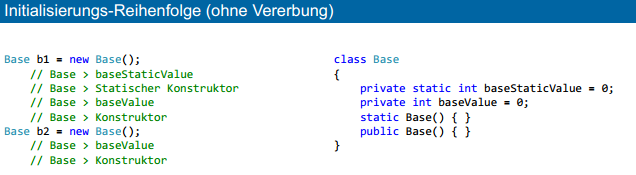


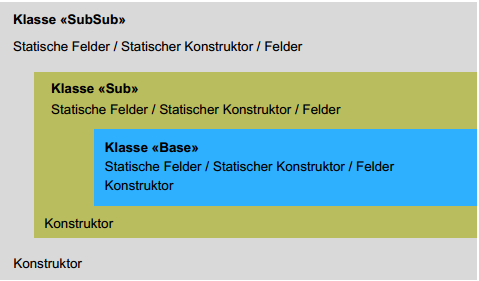
void Sum(out int sum, params int[] values)  
{  
 sum = 0;  
 foreach (int i in values) sum += i;  
}

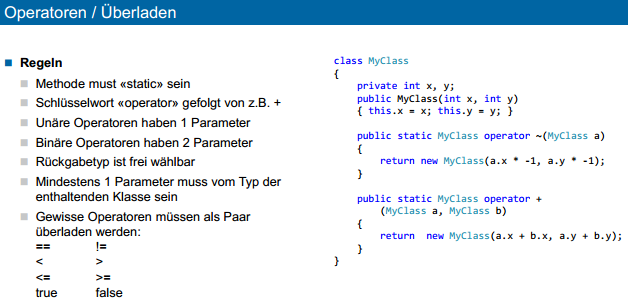
private void Sort(  
 int[] array, // Erforderlich  
 int from = 0, // Optional  
 int to = -1, // Optional  
 bool ascending = true, // Optional  
 bool ignoreCase = false // Optional  
)  
{ }  
void TestSortNamed()  
{  
 int[] a = { 3, 5, 2, 6, 8, 4 };  
 Sort(a, ascending: false);  
 Sort(a, ignoreCase: true, from: 3);  
 Sort(a, ignoreCase: true, ascending: false, to: 2, from: 3);  
 Sort(a, ignoreCase: true, ascending: false, from: 2, to: 3);  
 Sort(ignoreCase: true, ascending: false, from: 2, to: 3, array: a);  
}

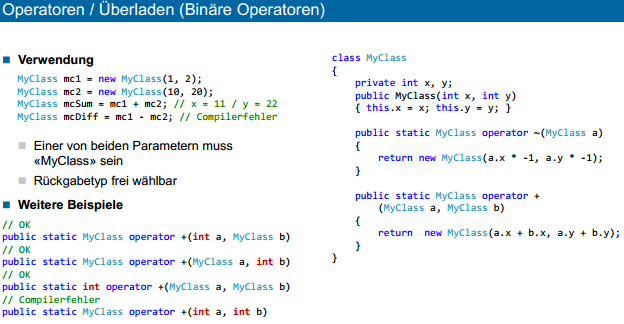


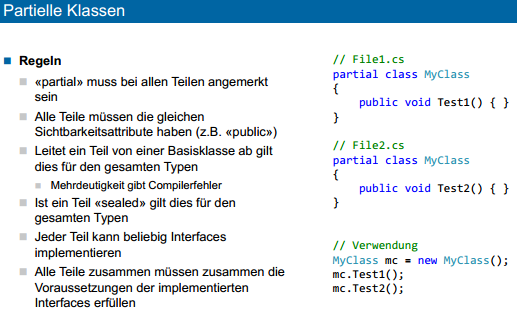


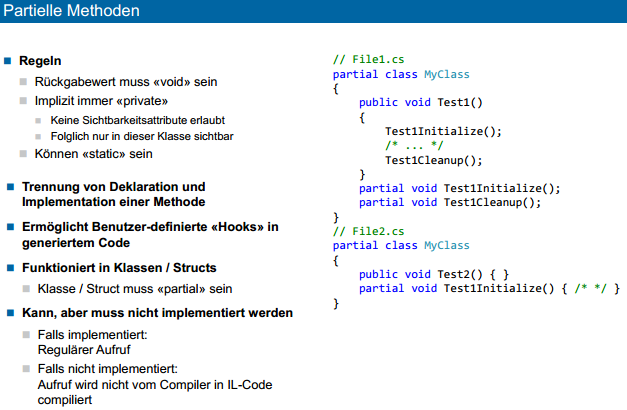






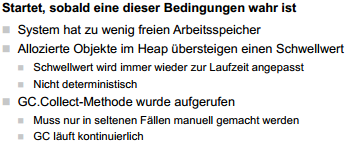




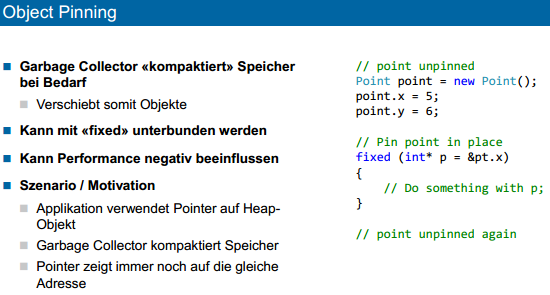


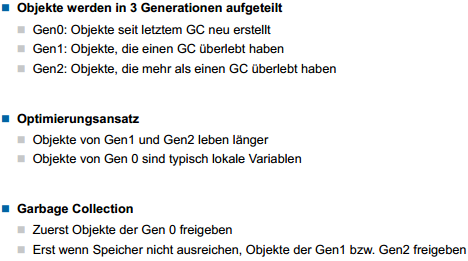
# GC, Destruktion & IDisposable

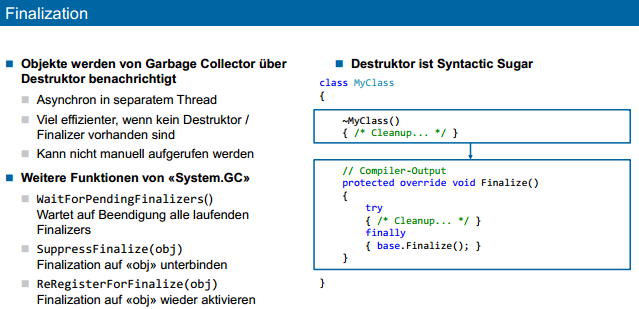
Mark & Sweep mit Generationen

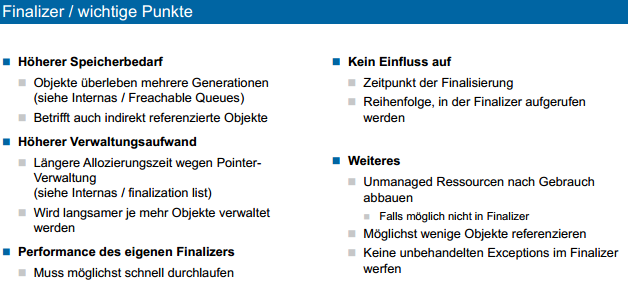


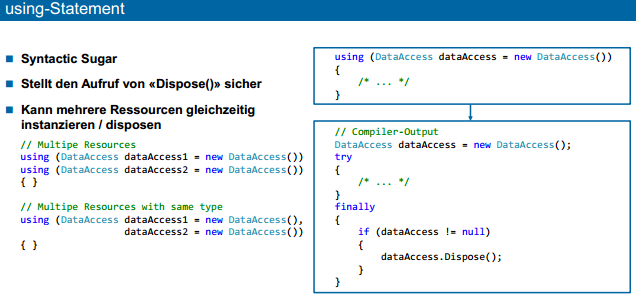
Zudem Large Object heap > 85'000 Bytes

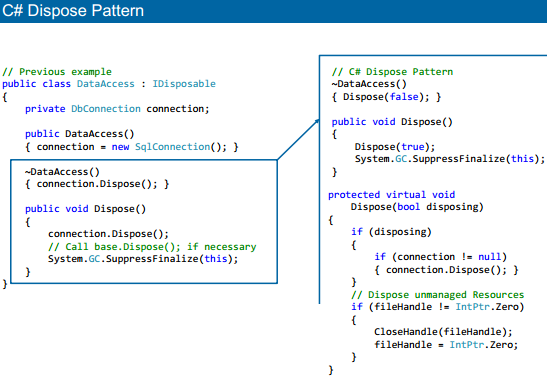


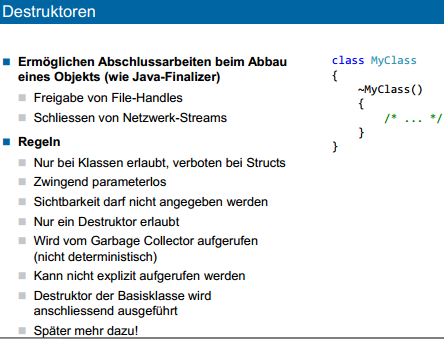


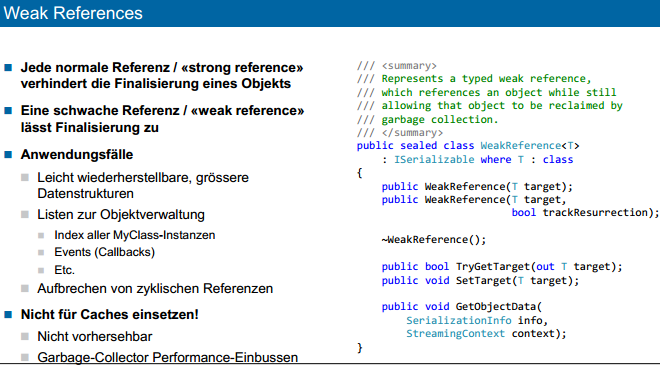




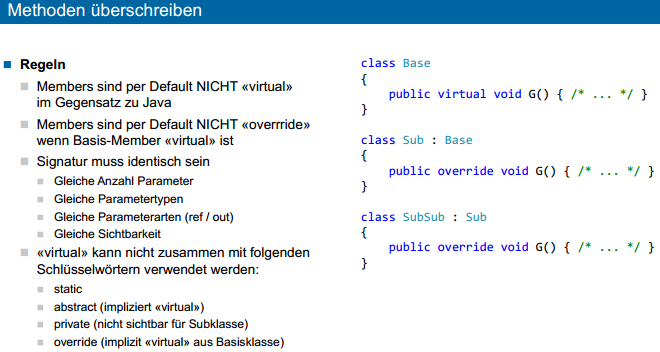


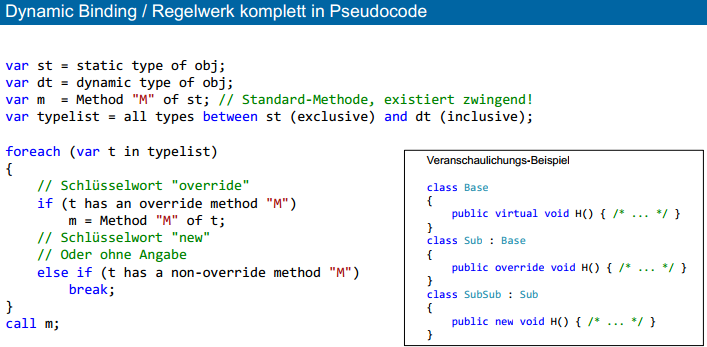


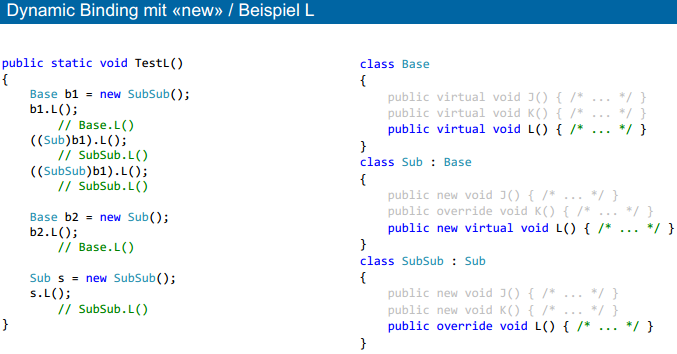


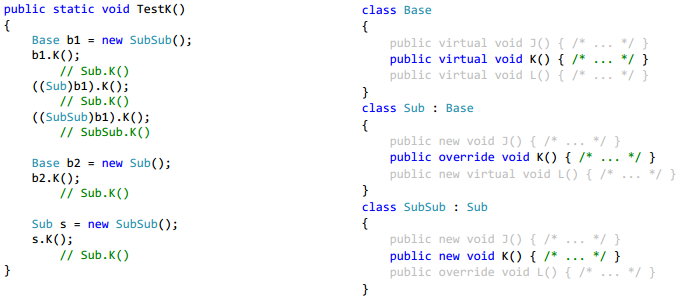


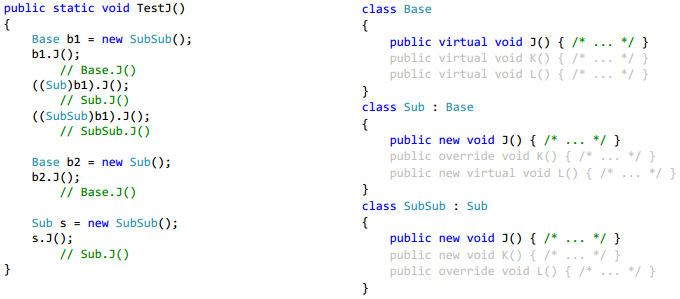
# Vererbung

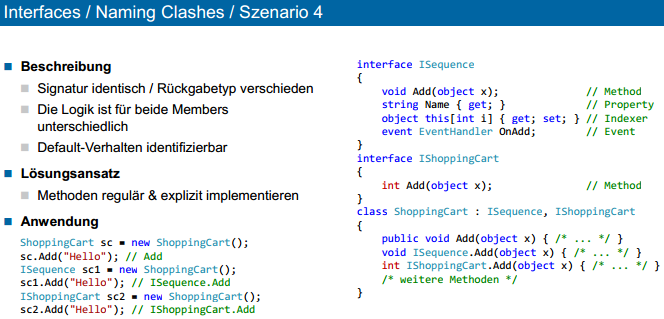


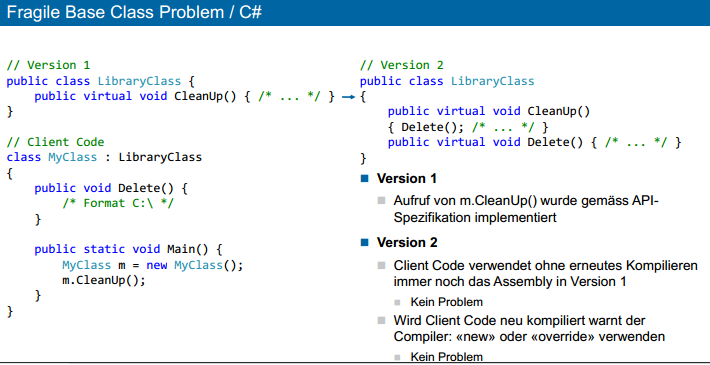






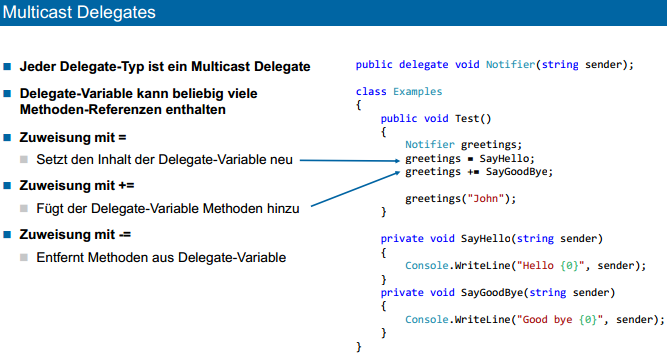


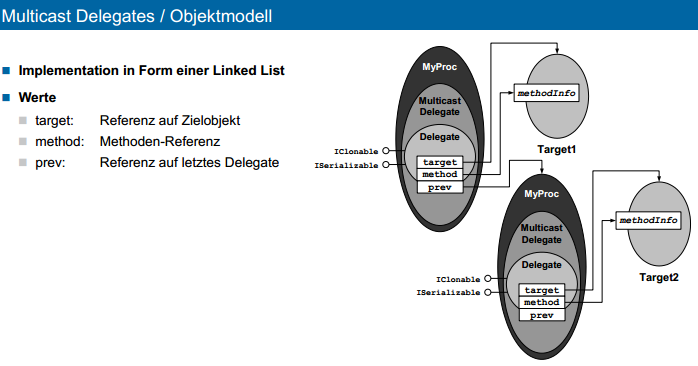


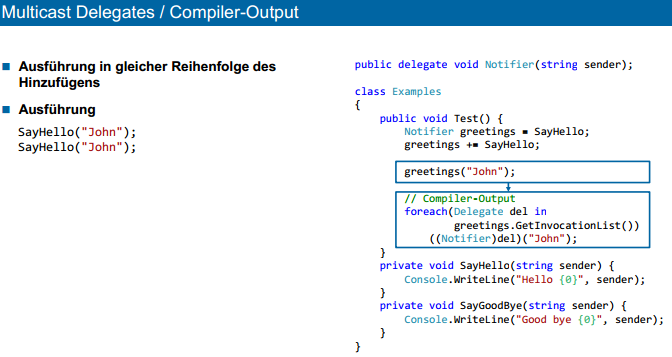


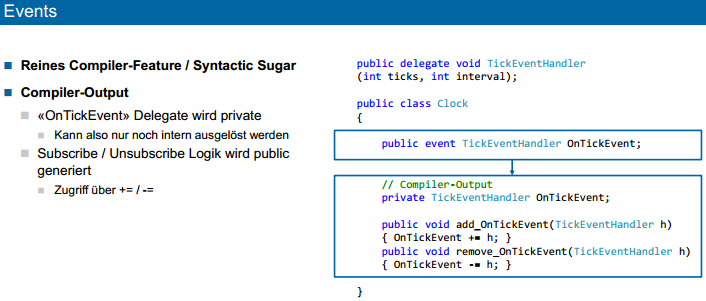
Java versagt, da dort standartmässig Dynamic Dispatch für alle Methoden ist.

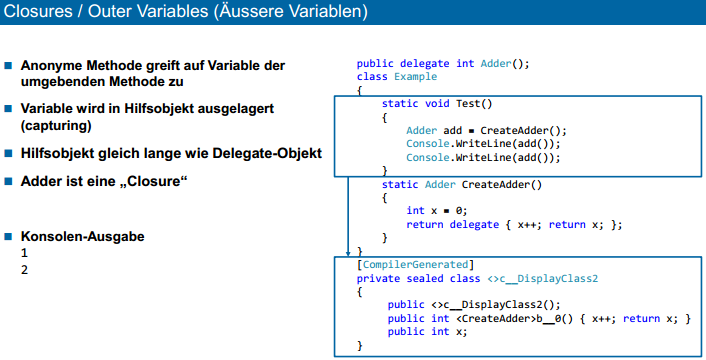
# Delegates & Events



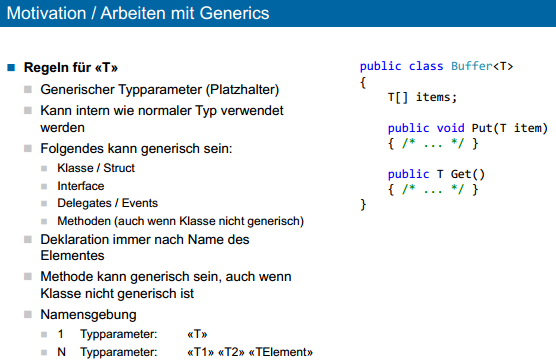


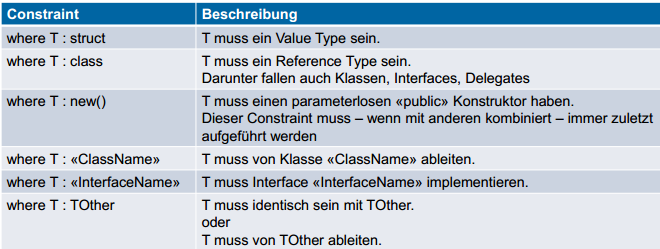


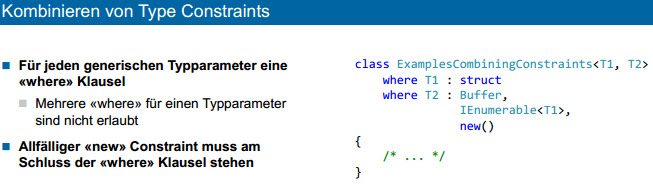


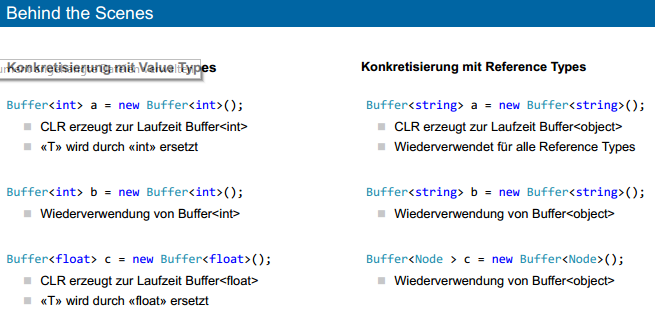


# Generics

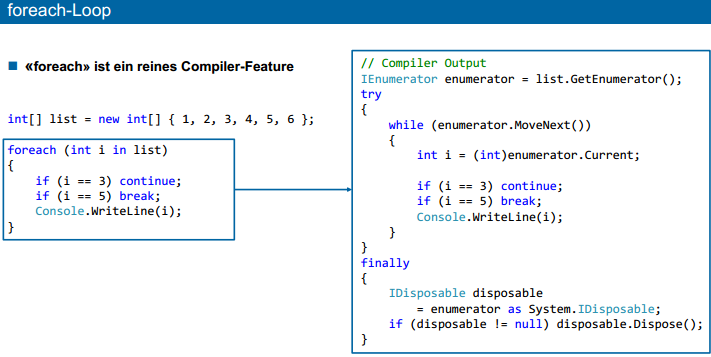


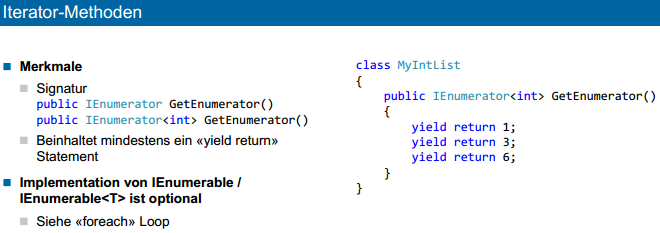


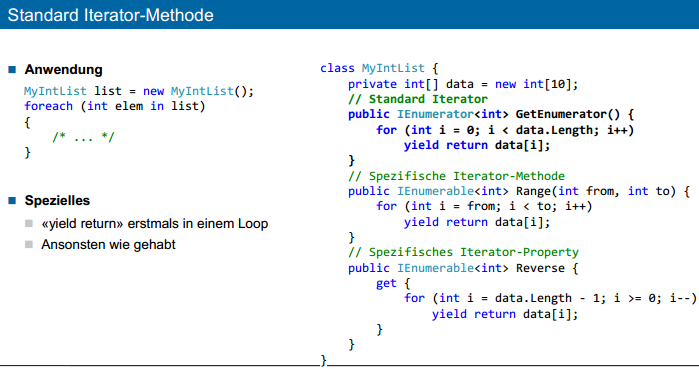




# Iteratoren







class MyClass {

int limit = 0;

public MyClass(int limit) { this.limit = limit; }

public IEnumerable<int> CountFrom(int start)

{

for (int i = start; i <= limit; i++) {

yield return i;

}

}

}

class MyClass\_Enumerator : IEnumerable<int> {

int state$0 = 0;// internal member

int current$0; // internal member

MyClass this$0; // implicit parameter to CountFrom

int start; // explicit parameter to CountFrom

int i; // local variable of CountFrom

public int Current {

get { return current$0; }

}

public bool MoveNext()

{

switch (state$0) {

case 0: goto resume$0;

case 1: goto resume$1;

case 2: return false;

}

resume$0:;

for (i = start; i <= this$0.limit; i++) {

current$0 = i;

state$0 = 1;

return true;

resume$1:;

}

state$0 = 2;

return false;

}

... other bookkeeping, not important here ...

}

public IEnumerable<int> CountFrom(int start)

{

MyClass\_Enumerator e = new MyClass\_Enumerator();

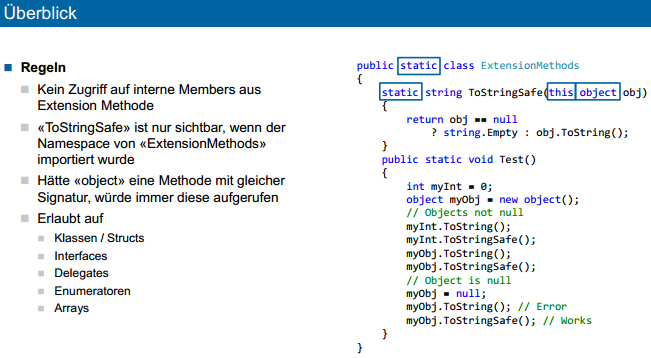
e.this$0 = this;

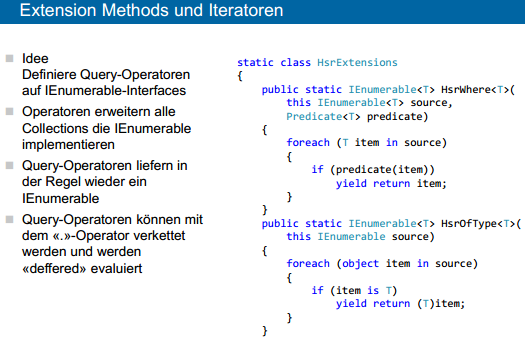
e.start = start;

return e;

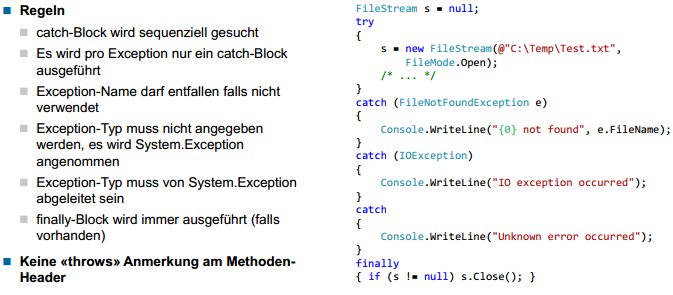
}

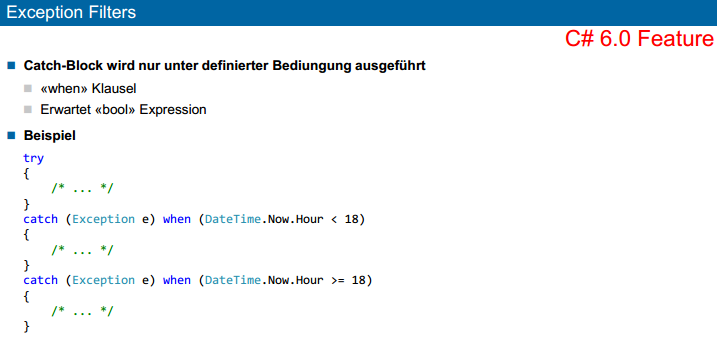
# Extension Methods



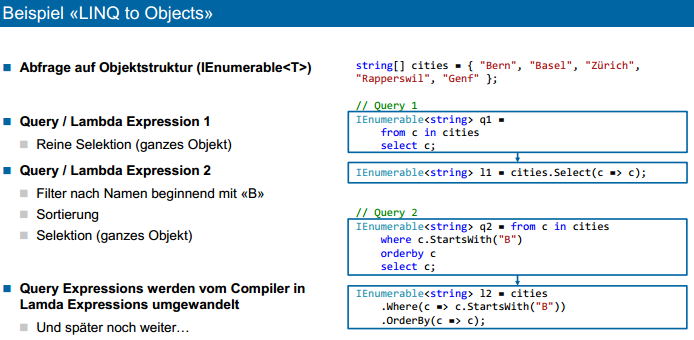


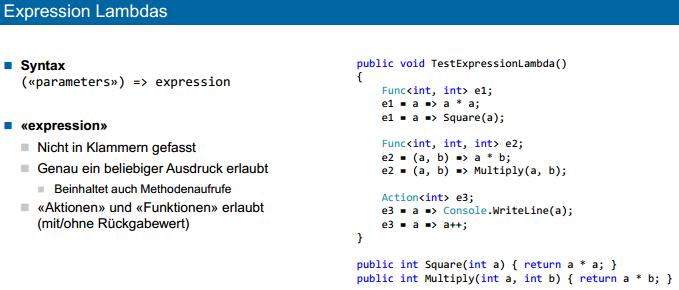
# Exceptions

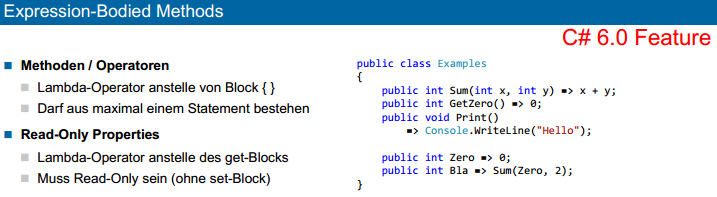


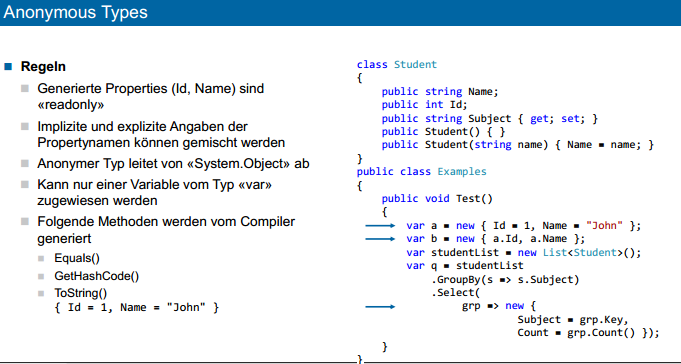


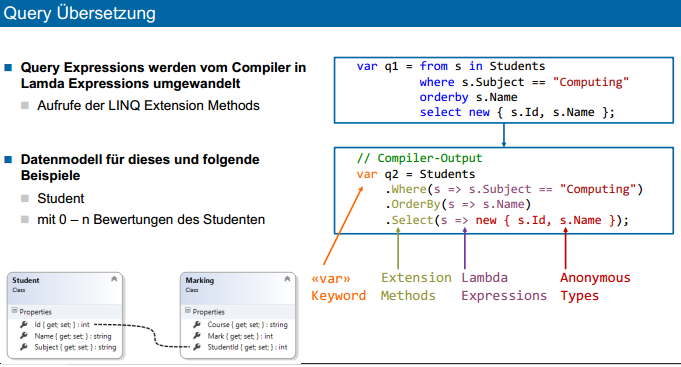
# LINQ

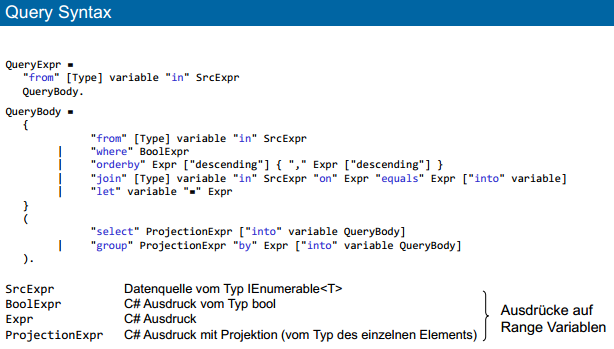


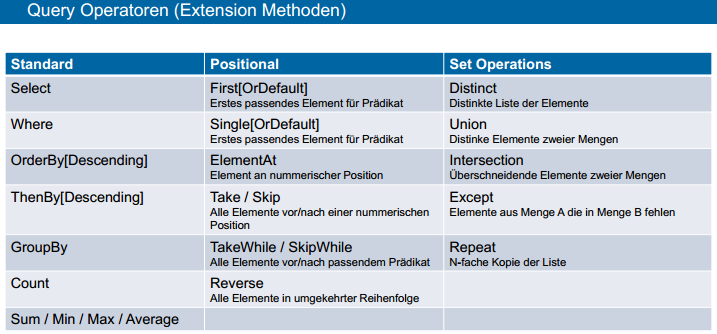


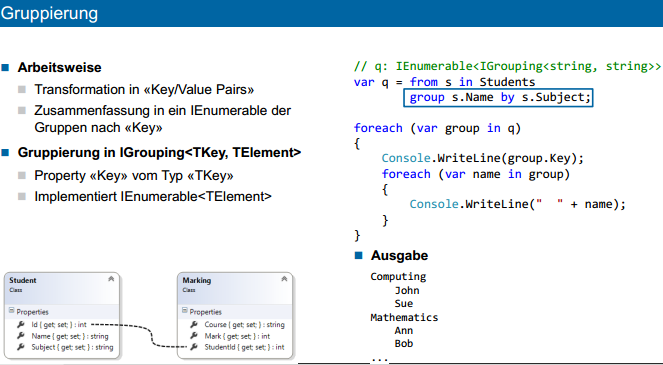


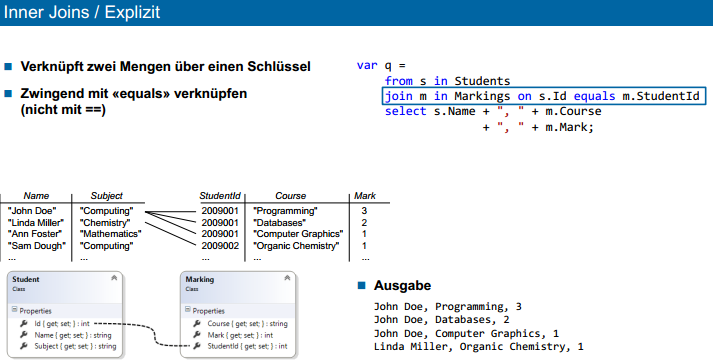


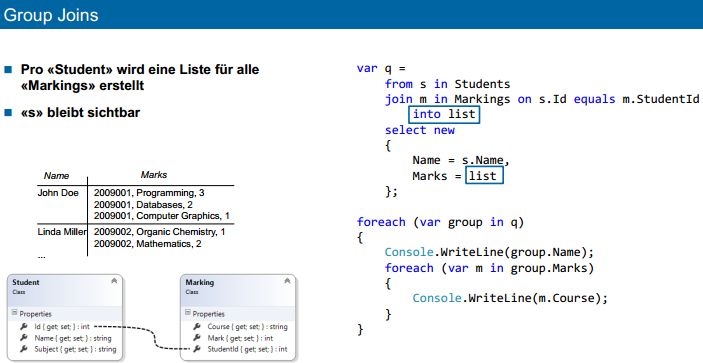


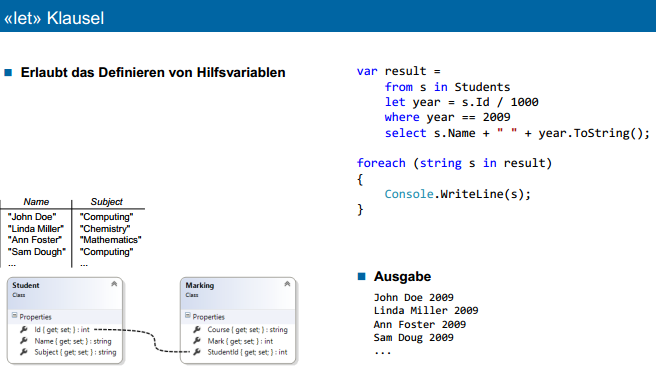


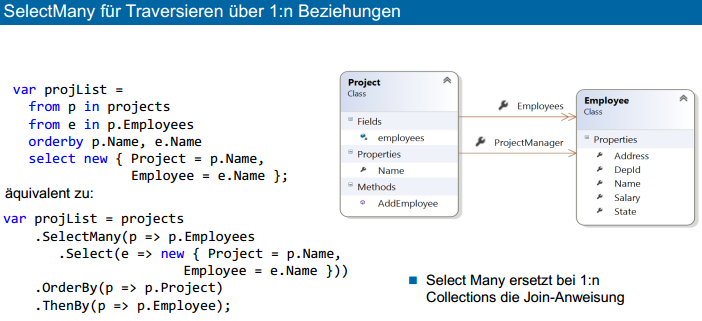




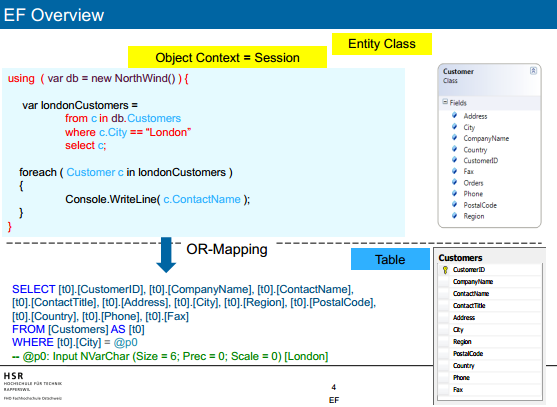


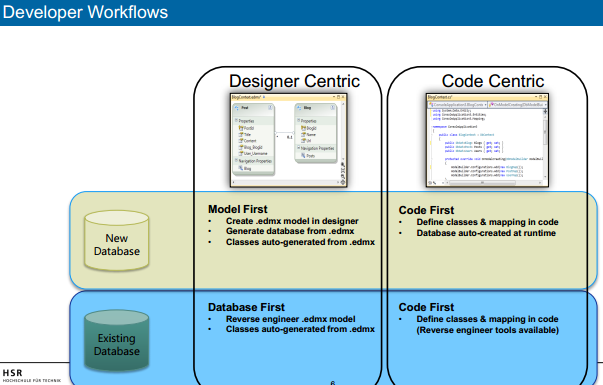


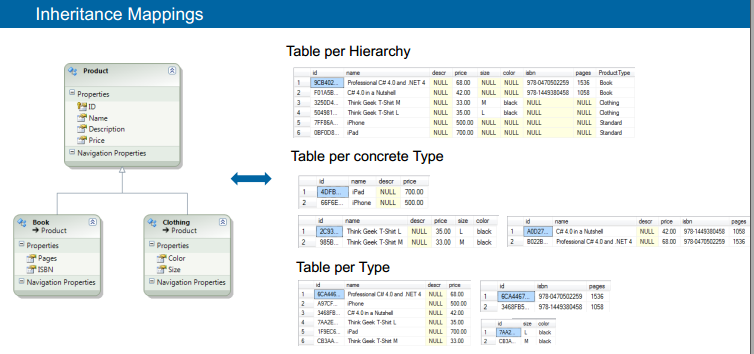


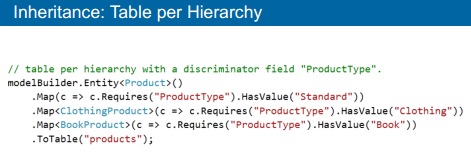


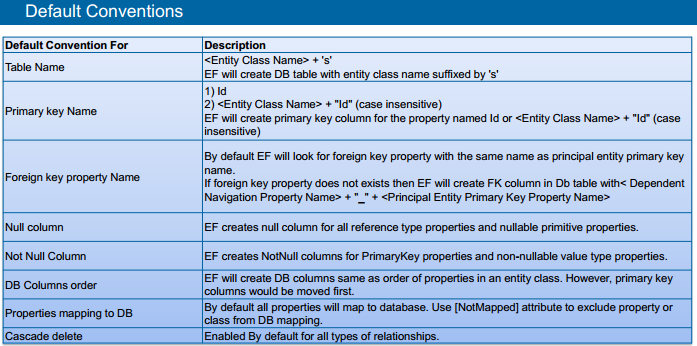
# Entity Framework

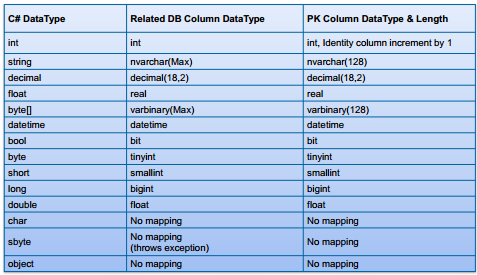


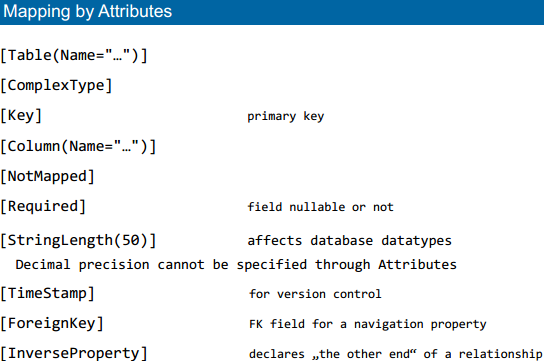


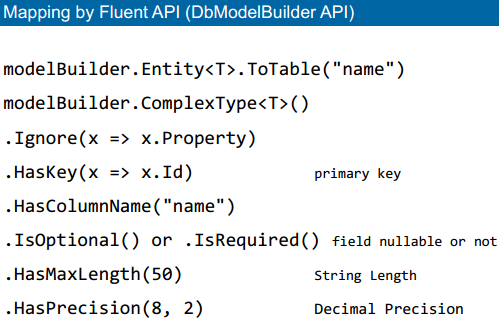




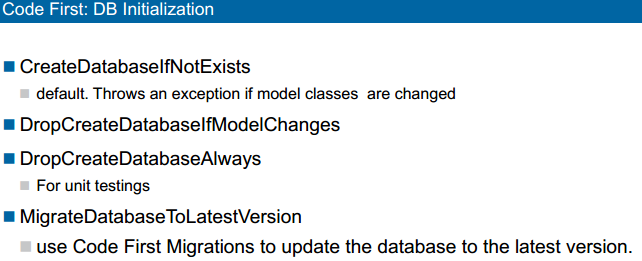


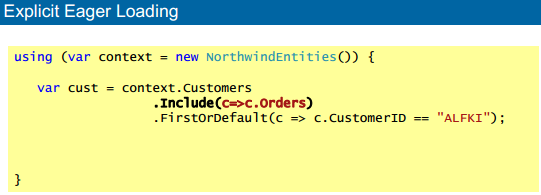


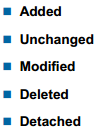
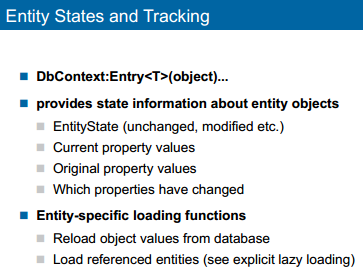


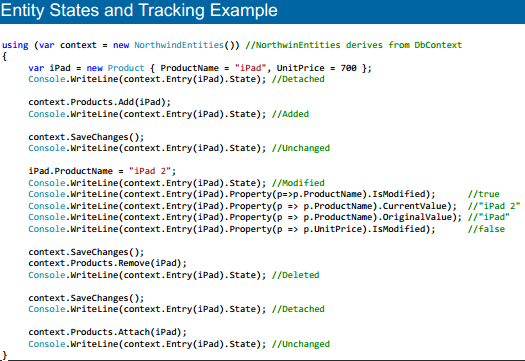


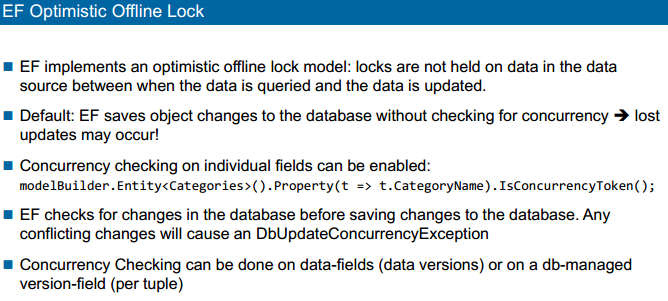


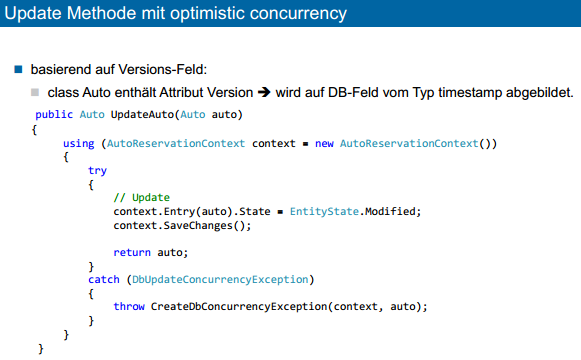


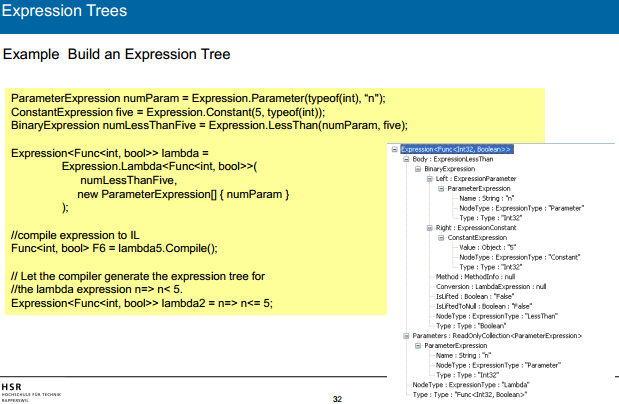




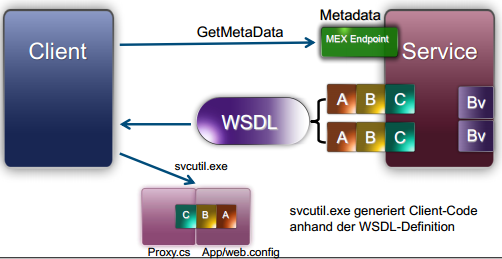
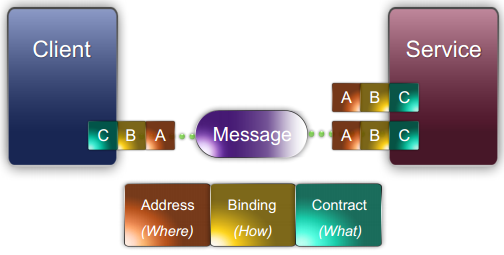


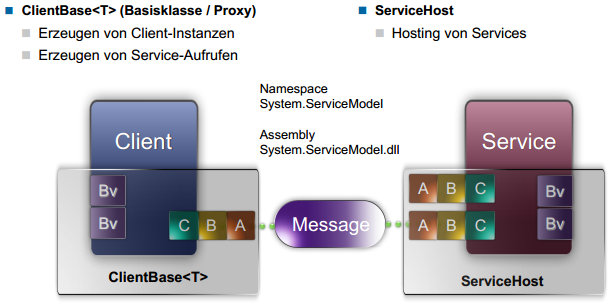




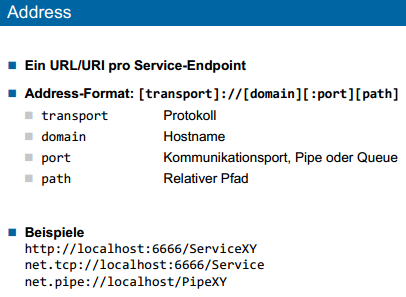


# WCF

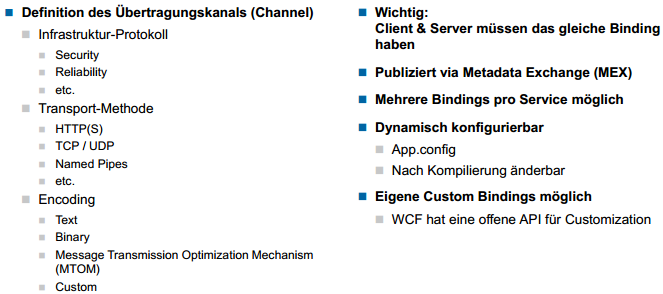


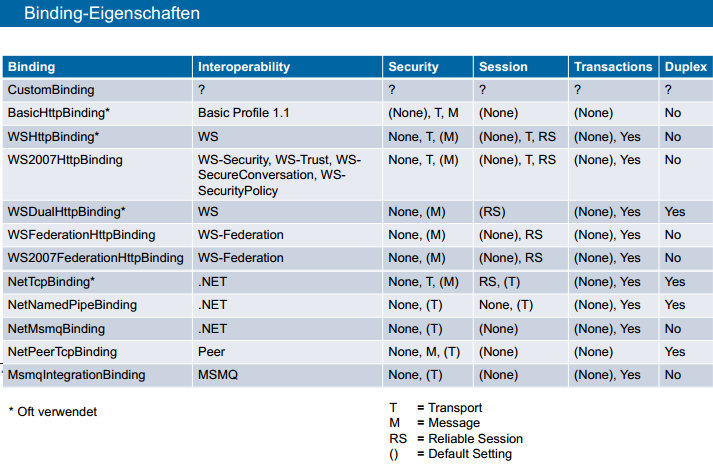


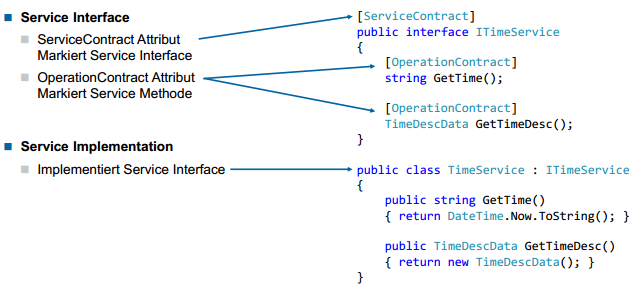


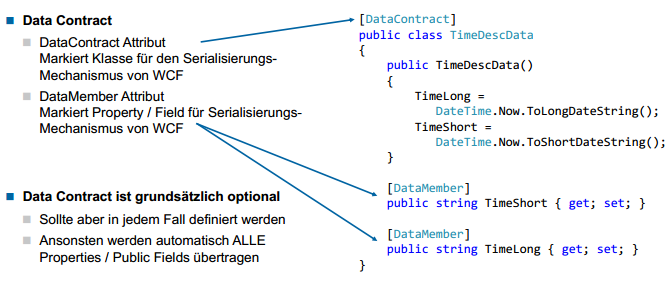


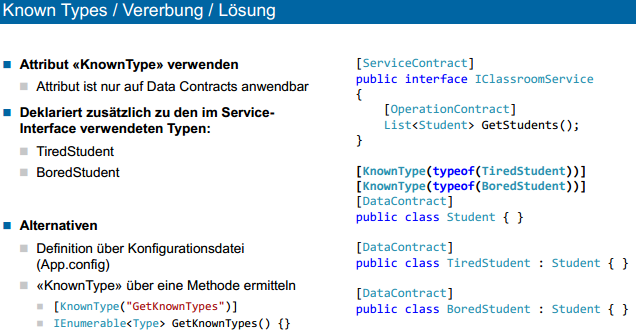
**Binding**

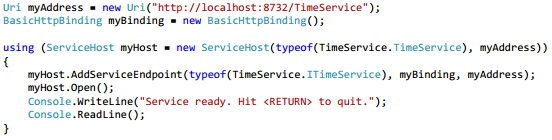


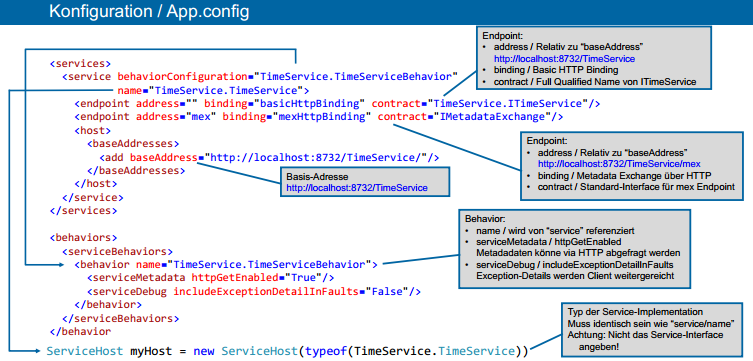


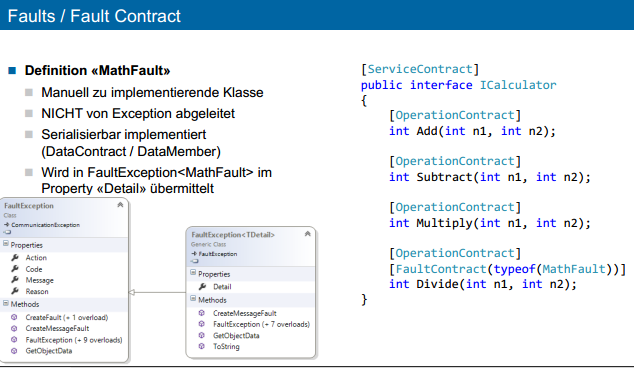


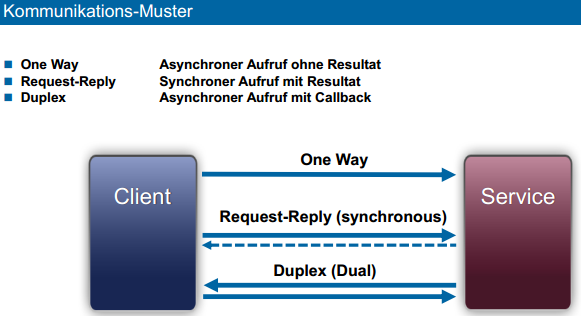


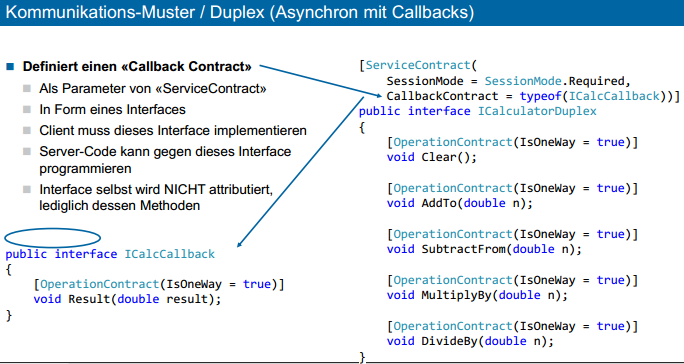




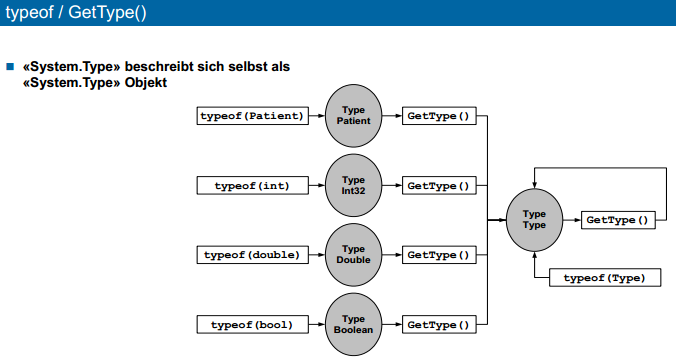


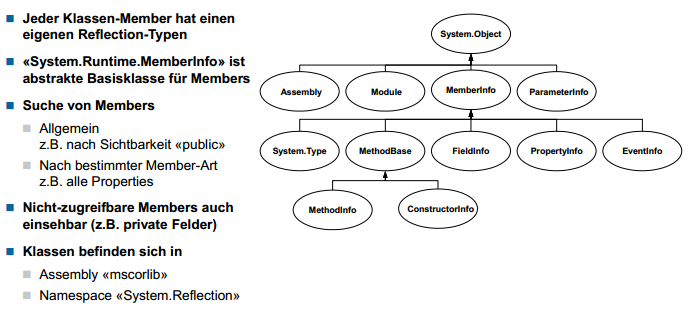


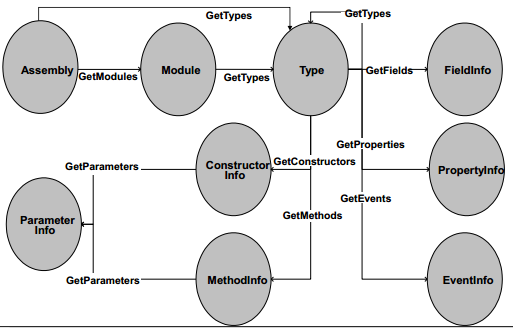




# Reflection







# Attributes

