05 – Methoden

Bildungszentrum Uster
Höhere Berufsbildung
Uster
HBU

Agenda

- Instanzmethoden
- Statische Methoden
- Überladene Methoden
- Überschriebene Methoden
- Named Arguments und Standardwerte
- Objektreferenzen als Parameter
- Methoden mit mehreren Rückgabewerten
- Erweiterungsmethoden

Bildungszentrum Uster
Höhere Berufsbildung Uster
HBU

Methoden

Methoden deklarieren

- Eine Methode beschreibt das Verhalten einer Klasse
- Syntax:

```
<Return Type> <Method Name> (<Parameter>, …)
{
  <Implementation>
}
```

- Beispiele (ohne Implementierung):

```
void GibAus() { … }

void GibAus(string text) { … }

int Addiere(int zahl1, int zahl2) { … }
```

Bildungszentrum Uster
Höhere Berufsbildung Uster
HBU

Methoden

Methoden deklarieren

- Expression-bodied-Syntax mit «=>» möglich, wenn der Methodenkörper nur aus einem Ausdruck besteht
- Syntax:

```
<Return Type> <Method Name> (<Parameter>, …)
    => <Ausdruck>
```

- Beispiel:

```
int Addiere(int zahl1, int zahl2)
    => return zahl1 + zahl 2;
```

Bildungszentrum Uster
Höhere Berufsbildung Uster
HBU

Methoden

Parameter

- Die Parameter bilden die Schnittstelle der Methode
- Parameterdeklaration beinhaltet Typ und Namen
- Parameter werden durch Komma getrennt
- Eine Methode kann 0..n Parameter haben
- Beispiel:

```
int Addiere(int zahl1, int zahl2)
{
    return zahl1 + zahl2;
}
```

Bildungszentrum Uster
Höhere Berufsbildung Uster
HBU

Variable Anzahl Parameter mit `params`

- Codebeispiel `Add()`

```
public static int Add(params int[] zahlen)
{
    int summe = 0;
    foreach (int zahl in zahlen)
    {
        summe += zahl; // summe = summe + zahl;
    }
    return summe;
}
```

Methoden

Rückgabewert

- Alle einfachen und zusammengesetzten Typen
- `void`, falls kein Rückgabewert gebraucht wird
- Rückgabewert wird mit `return` zurückgegeben
- Bei `void`-Methode entfällt das `return`-Ausdruck

Beispiel

```
int Addieren(int zahl1, int zahl2)
{
    return zahl1 + zahl2;
}
```

Bildungszentrum Uster
Höhere Berufsbildung Uster
HBU

Methoden

Mehrere `return`-Pfade

- Eine Methode kann mehr als einen `return`-Pfad haben

Codebeispiel

```
int BerechneMaximum(int zahl1, int zahl2)
{
    if (zahl1 > zahl2)
    {
        return zahl1;
    }
    else
    {
         return zahl2;
    }
}
```

Bildungszentrum Uster
Höhere Berufsbildung Uster
HBU

Methoden

Unerreichbarer Code

- Anweisung hinter einem `return`-Ausdruck
- Compiler-Warnung: “unreachable code detected”

- Beispiel:

```
int Addiere(int zahl1, int zahl2)
{
  return zahl1 + zahl2;

  // Unreachable code
  int summe;
  summe = zahl1 + zahl2;
}
```

Bildungszentrum Uster
Höhere Berufsbildung Uster
HBU

Statische Methoden

- Syntax:

```
static <Return Type> <Method Name> (<Parameter>, …)
{
      <Implementation>
}
```

- Methoden werden als statisch deklariert, wenn
  - die Methode einen generellen Charakter besitzt, z.B. Methoden in der Klasse `System.Math`,
  - keine Instanziierung eines Objektes benötigt wird oder
  - die benötigten Informationen aus den Übergabeparametern entnommen werden.

Statische Methoden

Beispiel - Statische Methoden in Klassen:

```
public class Formeln
{
  public static double KreisUmfang(double radius)
  {
    return (2* Math.PI * radius);
  }
}
```

Aufruf von statischen Methoden:

```
double umfang = Formeln.KreisUmfang(321.23);
Console.Write(umfang);
```

Bildungszentrum Uster
Höhere Berufsbildung Uster
HBU

Überladene Methoden

Methoden überladen

- Methoden mit semantisch identischen Aufgaben
- Name der Methode wird wiederverwendet
- Lesbarkeit des Quellcodes bleibt erhalten

- Unterschiedliche Parameterliste:
  - Unterschiedliche Anzahl Parameter
  - Verschiedene Typen in der Parameterliste

- Namen der Parameter haben keinen Einfluss
- Rückgabetyp hat keinen Einfluss

Bildungszentrum Uster
Höhere Berufsbildung Uster
HBU

Überladene Methoden

Beispiele überladener Methoden

```
// Bsp. 1
void GibAus(int zahl) { … };
void GibAus(int zahl1, int zahl2) { … };
void GibAus(string Text) { … };
void GibAus(float zahl) { … };
int GibAus(int zahl) { …; return xy; };

// Bsp. 2
float Addiere(float zahl1, float zahl2) { … };
int Addiere(int zahl1, int zahl2) { … };
```

✘ Fehler in Kombination mit Methode in 1. Zeile

```
int Addiere(int zahl1, int zahl2) { … };
int Addiere(int summand1, int summand2) { … };
```

✘ Fehler

Bildungszentrum Uster
Höhere Berufsbildung Uster
HBU

Überschriebene Methoden

Methoden überschreiben (engl. to override)

- Schlüsselwort `override` verwenden
- Beispiel Methode `ToString()` überladen

```
public class Employee
{
    public int Id { get; set; }
    public string FirstName { get; set; }
    public string LastName { get; set; }

    public override string ToString()
    {
        return $"{Id}: {LastName}, {FirstName}";
    }
}
```

Bildungszentrum Uster
Höhere Berufsbildung
Uster
HBU

Übung 5.1

Methoden überladen

- Schreibe die Klasse `Kalkulation`, welche mehrere `Add()`-Methoden enthält:
  - `float Add(int, int)`
  - `float Add(float, int)`
  - `float Add(float, float)`

Bildungszentrum Uster
Höhere Berufsbildung Uster
HBU

Named Arguments und Standardwerte

Named Arguments

- Named Arguments können beim Methodenaufruf verwendet werden
- Standardwerte werden bei der Methode dem Argument zugewiesen

```
static void Main(string[] args)
{
    double price = 80;
    double amount = GetPriceWithTransportTax(price);
    Console.WriteLine($"Preis: {price:C2}, mit Transportkosten: {amount:C2}");

    amount = GetPriceWithTransportTax(price, limit: 70);
    Console.WriteLine($"Preis: {price:C2}, mit Transportkosten: {amount:C2}");
}

static double GetPriceWithTransportTax(double price, double limit = 100)
{
    return price >= limit ? price : price + 10.5;
}
```

Anwendung Methodenaufruf

Standartwert am Argument zuweisen

Bildungszentrum Uster
Höhere Berufsbildung Uster
HBU

Objektreferenzen als Parameter

Codebeispiel (1/2)

- Vorsicht: Die Methode verändert das als Parameter referenzierte Objekt

```
static void Main(string[] args)
{
    EBike eBike = new EBike()
    {
        Id = 1,
        Brand = "PowerRacer",
        Model = "Storm 4",
        PurchasePrice = 2500,
        PurchaseDate = new DateTime(2016, 8, 1)
    };

    CalculateResalePrice(eBike);
    Console.WriteLine(eBike);
}

private static void CalculateResalePrice(EBike bike)
{
    TimeSpan timeSpan = DateTime.Now - bike.PurchaseDate;
    double minFactor = (timeSpan.TotalDays / 365) >= 5 ? 0 : (timeSpan.TotalDays / 365) * 0.2;
    double maxFactor = (timeSpan.TotalDays / 365) >= 5 ? 0 : (timeSpan.TotalDays / 365) * 0.15;
    bike.ResalePriceMin = bike.PurchasePrice * (1 - minFactor);
    bike.ResalePriceMax = bike.PurchasePrice * (1 - maxFactor);
}
```

Der Inhalt des Objektes eBike in Main() wird verändert

Objektreferenzen als Parameter

Codebeispiel (2/2)

- Klasse `EBike`

```
public class EBike
{
    public int Id { get; set; }
    public string Brand { get; set; }
    public string Model { get; set; }
    public double PurchasePrice { get; set; }
    public DateTime PurchaseDate { get; set; }
    public double ResalePriceMin { get; set; }
    public double ResalePriceMax { get; set; }

    public override string ToString()
    {
        return $"{Id}: {Brand}, {Model}, " +
            $"Purchase: {PurchaseDate.ToShortDateString()}, {PurchasePrice:C2}," +
            $"Resale [min:{ResalePriceMin:C2}, max:{ResalePriceMax:C2}] ";
    }
}
```

Bildungszentrum Uster
Höhere Berufsbildung
Uster
HBU

Methoden mit mehreren Rückgabewerten

Ausgangslage

- Mit `return` kann nur ein Wert zurückgegeben werden
- **Beispiel:** Umrechnung von Polar- in Karthesische-Koordinaten

```
? Polar2Karth(float radius, float winkel)
{
  // Hier soll umgerechnet werden
  // und die X/Y-Koordinaten zurückgegeben
  // werden
}
```

- **Frage:** Wie kann eine Methode mit mehreren Rückgabewerten implementiert werden?

Bildungszentrum Uster
Höhere Berufsbildung Uster
HBU

Methoden mit mehreren Rückgabewerten

1. Lösung: Strukturierter Datentyp

- Als Rückgabeparameter kann ein strukturierter Datentyp benutzt werden
- Beispiel:

```
public class Koordinaten
{
    public double X { get; set; }
    public double Y { get; set; }
}
```

```
public static Koordinaten PolarToKarthAsClass(double radius, double winkel)
{
    Koordinaten coord = new Koordinaten();
    coord.X = radius * Math.Cos(winkel);
    coord.Y = radius * Math.Sin(winkel);
    return coord;
}
```

Methoden mit mehreren Rückgabewerten

2. Lösung: Array

- Als Rückgabeparameter kann ein Array benutzt werden
- Beispiel:

```
public static double[] PolarToKarthAsArray(double radius, double winkel)
{
    double[] koordArray = new double[2];
    koordArray[0] = radius * Math.Cos(winkel);
    koordArray[1] = radius * Math.Sin(winkel);
    return koordArray;
}
```

Methoden mit mehreren Rückgabewerten

3. Lösung: Rückgabe mit `out`-Parametern

- Die Rückgabeparameter können mit dem Schlüsselwort `out` gekennzeichnet werden
- Jedem `out`-Paramater muss in der Methode einen Wert zugewiesen werden, da sie in der aufrufenden Methode nicht zwingend initialisiert werden müssen

```
public static void PolarToKarthAsOutParam(
    double radius, double winkel, out double x, out double y)
{
    x = radius * Math.Cos(winkel);
    y = radius * Math.Sin(winkel);
}
```

Bildungszentrum Uster
Höhere Berufsbildung Uster
HBU

Methoden mit mehreren Rückgabewerten

3. Lösung: Rückgabe mit `out`-Parametern

- Beim Methoden Aufruf müssen die Parameter mit `out` gekennzeichnet werden
- Beispiel:

```
static void TextMultiParameterAsOutParam()
{
    double radius = 21.2;
    double winkel = 1.23;
    double x, y;
    ConvertHelper.PolarToKarthAsOutParam(radius, winkel, out x, out y);
}
```

Methoden mit mehreren Rückgabewerten

4. Lösung: Rückgabe mit `ref`-Parametern

- Der Methode werden Referenzen auf die Variablen übergeben
- Vor dem Methodenaufruf müssen die Variablen Verwendung initialisiert sein!
- Beispiel:

```
static void TextMultiParameterAsRefParam()
{
    double radius = 21.2;
    double winkel = 1.23;
    double x=0, y=0;
    ConvertHelper.PolarToKarthAsRefParam(radius, winkel, ref x, ref y);
}
```

Bildungszentrum Uster Höhere Berufsbildung Uster HBU

Methoden mit mehreren Rückgabewerten

Beispiel mit `ref`-Parametern

```
static void TextMultiParameterAsRefParam()
{
    double radius = 21.2;
    double winkel = 1.23;
    double x=0, y=0;
    ConvertHelper.PolarToKarthAsRefParam(radius, winkel, ref x, ref y);
}
```

```
public static void PolarToKarthAsRefParam(
    double radius, double winkel, ref double x, ref double y)
{
    x = radius * Math.Cos(winkel);
    y = radius * Math.Sin(winkel);
}
```

Bildungszentrum Uster
Höhere Berufsbildung Uster
HBU

Methoden mit mehreren Rückgabewerten

Gegenüberstellung: Call by Value und Call by Refernce

Call by Value:

- Argumente werden kopiert
- Argument unverändert
- Keine Seiteneffekte
- Kopieren braucht Zeit

- Default

Call by Reference

- Referenz auf Argument verwendet
- Argument verändert
- Seiteneffekte oft gewünscht
- Performanz optimiert

- `ref`-Schlüsselwort

Übung 5.2

`out`-Parameter

- 1) Schreibe eine Klasse `Converter` mit Methoden, um Koordinaten vom Polarformat in karthesische Koordination zu konvertieren, sowie in die andere Richtung. Verwende `out` oder `ref`, um mehrere Parameter zurückzugeben.
- 2) Schreibe eine Klasse `Testtreiber`, um die obigen Methoden zu testen.

Bildungszentrum Uster
Höhere Berufsbildung Uster
HBU

Erweiterungsmethoden

Einleitung

- Erweiterungsmethoden (extension methods)…
  - Ermöglichen die Erweiterung einer bestehenden Klasse ohne eine neue Subklasse mit Vererbung zu implementieren
  - Werden als statische Methoden in einer separaten statischen Klasse implementiert
  - Sind in der Anwendung gleich wie nicht-statische Methoden; sie werden also auf einem konkreten Objekt angewandt
  - Definieren als erstes Argument eine Referenz `this` auf das angewandte konkrete Objekt
  - Die meisten LINQ-Standartoperatoren sind als Erweiterungsmethoden implementiert

Bildungszentrum Uster
Höhere Berufsbildung Uster
HBU

Erweiterungsmethoden

Beispiel-Implementierung

- Extension Method zur Erweiterung der Klasse `String` mit der Methode `WordCount()`

```
public static class MyExtensionMethods
{
    public static int WordCount(this String str)
    {
        return str.Split(new char[] { ' ', ',', ';', '.', '!', '?' },
            StringSplitOptions.RemoveEmptyEntries).Length;
    }
}
```

- Anwendung der Erweiterungsmethode `WordCount()`

```
string text = "Extension Methods sind ein tolles Feature.";
Console.WriteLine("Anzahl Wörter im Text: {0}", text.WordCount());
```

Bildungszentrum Uster
Höhere Berufsbildung Uster
HBU

Erweiterungsmethoden

Gut zu wissen

- Erweiterungsmethoden sollten nur implementiert werden, wenn es gute Gründe dafür gibt. Es wird Empfohlen die Erweiterungen normalerweise mittels Vererbung zur implementieren.
- Eine bestehende Methode kann nicht durch eine Erweiterungsmethode mit gleicher Signatur überschrieben werden.
- Erweiterungsmethoden werden über den Namensraum eingebunden, in welchem diese implementiert wurden:

```
using MyExtensionsNamespace;
```

Bildungszentrum Uster
Höhere Berufsbildung
Uster
HBU

Übung 5.3

Erweiterungsmethoden

- Schreibe eine Erweiterungsmethode für die Klasse `Double`, um die Anzahl Dezimalstellen zu ermitteln.

Bildungszentrum Uster
Höhere Berufsbildung Uster
HBU