C#

Eigenschaften (Properties)

Properties

- Was ist eine Property?
- Wie funktioniert eine Property?

Properties

Was ist eine Property?

- Mit Properties kann man Klassen sichere und gekapselte Variablen geben.
- Properties werden verwendet, wenn man Variablen definieren will, die ein Objekt später beschreiben

Properties

```
class Person
{
    //Eigenschaften
    public string Name { get; set; }
    public int Alter { get; set; }
    public float Größe { get; set; }
}
```

Properties

```
class Auto
{
    //Variablen
    private bool motorAn;
    private bool scheibenwischerAn;

    //Eigenschaften
    public string Farbe { get; set; }
    public int PS { get; set; }
    public int AnzahlTüren { get; set; }
}
```

Properties

Wichtig!

- Properties sind keine einfachen Variablen!!!
- Sie sind ein Konstrukt aus 2 Methoden und einer Variablen.
- Eine Property hat eine set- und eine get-Methode.
- Zu einer Property gehört eine private Variable.

Properties

```
class Auto
{
    //Variablen
    private bool motorAn;
    private bool scheibenwischerAn;

    //Eigenschaften
    public string Farbe { get; set; }
    public int PS { get; set; }
    public int AnzahlTüren { get; set; }
}
```

```
class Program
{
    static void Main(string[] args)
    {
        Auto car = new Auto();

        car.Farbe = "Grün";
        Console.WriteLine(car.Farbe);

        Console.ReadKey();
    }
}
```

Properties

Was ist Get und Set?

- Die Methode Get dient zum Lesen der zu der Property gehörenden privaten Variablen.
- Die Methode Set dient zum Überscheiben der zu der Property gehörenden privaten Variablen.
- Über die Set und Get-Methode wird die private Variable gekapselt.

Automatische Properties

```
class Person
{
    //Eigenschaften
    public string Name { get; set; }
    public int Alter { get; set; }
    public float Größe { get; set; }
}
```

- Verwendung von automatisch implementierten Properties
- Es existiert **keine** private Variable hinter der Property

Properties

```
class Person
{
    //Private Variablen
    private int alter;

    //Eigenschaften
    public int Alter
    {
        get
        {
            Console.WriteLine("Alter wurde gelesen");
            return alter;
        }
        set
        {
            Console.WriteLine("Alter wurde überschrieben");
            alter = value;
        }
    }
}
```

```
class Program
{
    static void Main(string[] args)
    {
        Person peter = new Person();
        peter.Alter = 10;
        Console.WriteLine(peter.Alter);

        Console.ReadKey();
    }
}
```

Die Set-Methode beinhaltet häufig Plausibilitätsprüfungen.

Properties

```
1  public class Person{
2    private int alter;
3
4    public int Alter;
5    {
6      get
7      {
8        return alter;
9      }
10     set
11     {
12       if (value > 0)
13       {
14          alter = value;
15       }
16     }
17   }
18 }
```

Die Get-Methode liefert den Wert der privaten Variablen.

Die Set-Methode beinhaltet häufig Plausibilitätsprüfungen.

Achtung: **stack overflow**
da Set-Methode rekursiv aufgerufen wird

Lösung: Name in name ändern

```
class Person
{
    private string name;
    3 Verweise
    public string Name
    {
        set {
            if (value[0]=='S')
            {
                Name = value;
            }
        }
        get { return name; }
    }
}
```

```
class Program
{
    0 Verweise
    static void Main(string[] args)
    {
        Person p = new Person();
        p.Name = "Schicha";
        Console.WriteLine(p.Name);
    }
}
```