Cheatography

C# Cheat Sheet

by Netari via cheatography.com/30244/cs/8986/

Sichtbarkeiten Klasse

Klassen	
public	Unbeschränkt. Auch von anderen Assemblierungen aus können Objekte der Klasse erstellt werden.
internal	Nur innerhalb des aktuellen Projekts. Außerhalb des Projekts ist kein Objekt dieser Klasse erstellbar. Gilt als Standard, falls kein Modifizierer vorangestellt wird.
private	Nur innerhalb einer anderen Klasse. (Siehe Code)

```
class ParentClass
{
  private class ChildClass
  {
    public ChildClass()
   {
     Console.WriteLine("ChildClass");
   }
  }
}
```

Nullable Types

```
int ? x = null
```

int ? ermöglicht null Zuweisung

```
int y = x ?? 0;
```

Initialisiert y mit 0, wenn x ist null

Benannte Parameter

USE	Vereinfacht den Aufruf von mehreren optionalen Parametern
BSP	`add(string title, string autor ="")`
	`add(autor: "test", title: "test");`

Objektinitialisierer

```
Person pers1 = new Person { Name = "Bauer", Ort =
"Regensburg" }
```

Das Erzeugen und Initialisieren einer Instanz bedarf keinen Konstruktur.

Getter & Setter

```
private string _anrede;
public string anrede
{
    get { return(_anrede); }
    set { _anrede = value; }
}
```

Sichtbarkeiten Members

public	Unbeschränkt.
protected	Innerhalb der Klasse und der daraus abgeleiteten Klassen.
internal	Innerhalb des aktuellen Projekts.
internal protected	Innerhalb des aktuellen Projekts oder der abgeleiteten Klassen.
private	Nur innerhalb der Klasse.

Typinferenz (var)

EXP	Datentyp wird vom Compiler einmalig festgelegt und kann danach nicht mehr verändert werden.
USE	Referenzspeicherung eines Objekts (Anonymous Type)

```
var b = 7; // Integer
b = 12.3 // Fehler
var XMLSerializer = new XMLSerializer()
```

Parameterübergabe ref

EXP	Parameter werden als Referenz übergeben. (Zeiger)
BSP	`test(ref variable)`
USE	Nur wenn es sein muss

1. ref Parameter muss vorher initialisiert worden sein
2. Beim Aufruf muss ebenfalls das ref-Schlüsselwort vorangestellt werden.

Parameterübergabe out

Einziger Unterschied zwischen *ref* und *out* ist, dass eine Methode einem *ref*-Parameter einen Wert zuweisen **kann**, einem *out*-Parameter hingegen einen Wert zuweisen **muss**.

Objekt FAQ

Wie Zerstöre ich ein Objekt?

```
kunde1 = null;
```

Binding Property

```
public class Person
public String Anrede {get;set;}
private void Form1_Load(object sender, EventArgs e)
{
    textBox1.DataBindings.Add("Text", this,
"Anrede");
}
```