KLASSENATTRIBUTE UND KLASSENMETHODEN

Vincent Uhlmann
IT-Akademie Dr. Heuer GmbH

INSTANZATTRIBUTE VS. KLASSENATTRIBUTE

- **Instanzattribute** gehören zu einer **Instanz** der Klasse (also zu einem spezifischen Objekt)
- Jedes Objekt hat seine eigenen Werte für diese Attribute
- Beispiel: Jedes Objekt der Klasse Car hat ein eigenes Attribut Farbe

- **Klassenattribute (static)** gehören zur **Klasse selbst,** nicht zu einzelnen Objekten
- Alle Objekte teilen sich das gleiche Klassenattribut
- Beispiel: Ein Klassenattribut NumberOfCars, das für alle Objekte der Klasse Car gilt

KLASSENATTRIBUTE

- Gemeinsame Variable für alle Objekte
- Verwendung, um z.B. die Anzahl von Objekten eins Typs zu speichern
- Klassenweite Variable mit dem Schlüsselwort static markieren
- Die Variable existiert genau einmal, unabhängig von der Anzahl der Objekte

```
public class Car
{
    private static int _numberOfCars;
}
```

KLASSENATTRIBUTE

- Aufruf über Klassenname
- Klassenattribute können nur über den **Namen der Klasse** aufgerufen werden
- Beispiel: Car.NumberOfCars

```
public class Car
{
    public static int NumberOfCars;
}


int count = Car.NumberOfCars;
```

KLASSENMETHODEN

- Können aufgerufen werden, ohne dass eine Instanz der Klasse erstellt wird
- Haben **keinen** Zugriff auf Instanzattribute, da sie nicht an ein spezifisches Objekt gebunden sind
- Haben Zugriff auf Klassenattribute und andere Klassenmethoden

```
public class Car
{
    private static int _numberOfCars;

    public static int GetNumberOfCars()
    {
        return _numberOfCars;
    }
}
```

KLASSENMETHODEN

- Verwendung in eingebauten Klassen und Methoden des .NET-Frameworks
- Beispiele
  - Console.ReadLine()
  - Console.WriteLine()
  - Console.Write()
  - Math.Pow()
  - Math.Sqrt()
  - …

```
Console.WriteLine("Hallo");
```

STATISCHE KLASSEN

- **Statische Klassen** können nur **statische Mitglieder** enthalten (d.h. Methoden, Felder und Eigenschaften)
- Es können **keine Instanzen** einer statischen Klasse erstellt werden
- **Kein Konstruktor** für statische Klassen erlaubt (außer einem privaten statischen Konstruktor)
- Werden verwendet, wenn der Code keine Zustände speichern muss
- Werden für Utility-Methoden wie z.B. Mathematische Berechnungen, Formatierungen von Daten etc. genutzt

STATISCHE KLASSEN

```
public static class MathHelper
{
    public static double Add(double a, double b)
    {
        return a + b;
    }

    public static double Multiply(double a, double b)
    {
        return a * b;
    }
}
```

CONST

- Das Schlüsselwort const wird verwendet, um einen konstanten, unveränderlichen Wert zu definieren
- Wert muss zur Kompilierzeit bekannt sein
- Ist automatisch "static", alle Instanzen der Klasse haben denselben Konstanten Wert
- Wird vor allem für **Konstanten** genutzt, deren Wert sich **nie ändert** (z.B. mathematische Konstanten oder feste Werte)

```
public class Person
{
    public const int MaxNameLength = 50;
}
```

READONLY

- Wert kann entweder **bei der Deklaration** oder im **Konstruktor** festgelegt werden
- Kann für **Instanzattribute** oder **statische Attribute** genutzt werden
- Wert ist zur Laufzeit änderbar (beim Erstellen des Objekts oder im statischen Konstruktor), aber nach der Initialisierung nicht mehr veränderbar
- Ideal für Werte, die **einmalig festgelegt** werden müssen, aber erst zur **Laufzeit** verfügbar sind (z.B. IDs, die bei der Instanziierung erzeugt werden)

```
public class Gambler
{
    private static readonly Random _random = new Random();
}
```