# RECORD TYPE

Vincent Uhlmann
IT-Akademie Dr. Heuer GmbH

# WAS SIND RECORD TYPES

- Record Types sind eine spezielle Art von Klassen, die seit C# 9.0 verfügbar sind
- Sie bieten eine einfache Möglichkeit, unveränderliche Objekte zu definieren, die auf Wertebasis verglichen werden
- Record Types sind unveränderlich, was bedeutet, dass ihre Zustände nach der Erstellung nicht mehr geändert werden können
- Im Gegensatz zu Klassen, die auf Referenzbasis verglichen werden, werden Record Types auf Wertbasis verglichen
- Zwei Record-Objekte sind gleich, wenn alle ihre Eigenschaften gleich sind

# WOFÜR WERDEN SIE GENUTZT

- Record Types sind ideal für die Definition von einfachen Datenstrukturen, die unveränderlich sein sollen
- Sie ermöglichen eine einfache und intuitive Vergleichslogik, da sie auf Wertbasis verglichen werden
- Durch die Verwendung von Record Types kann der Boilerplate-Code für unveränderliche Datenstrukturen reduziert werden, da viele Methoden automatisch generiert werden

# SYNTAX

- Ein Record Type wird mit dem Schlüsselwort record definiert
- Die zu erstellenden Eigenschaften werden als Parameter deklariert

```
public record Person(string Name, int Age);
```

- Record Types können erweitert werden, um zusätzliche Methoden oder Eigenschaften hinzuzufügen

```
public record Person(string Name, int Age)
{
    public string GetFullName() => $"{Name} {Age}";
}
```

# SYNTAX

- Es können auch Record Types mit änderbaren Eigenschaften erstellt werden

```
public record Person
{
    public string Name { get; set; }
    public int Age { get; set; }
}
```

# INTEGRIERTE FORMATIERUNG

- Record Types verfügen über eine vom Compiler generierte ToString-Methode, die die Namen und Werte der öffentlichen Eigenschaften und Felder anzeigt.
- Die ToString-Methode gibt eine Zeichenfolge in folgendem Format zurück

```
<Datensatztypname> { <Eigenschaftsname> = <Wert>, <Eigenschaftsname> = <Wert>, ...}
```

```
public record Person(string Name, int Age);

Person person1 = new Person("Max", 30);

Console.WriteLine(person1); // Ausgabe: Person { Name = Max, Age = 30 }
```

# VERERBUNG VON RECORD TYPES

- Record Types können von anderen Record Types erben
- Ein Record kann nicht von einer Klasse erben, und eine Klasse kann nicht von einem Record erben
- Die Syntax für die Vererbung von Record Types ist ähnlich wie bei der Vererbung von Klassen
- Record Types können als abstract, sealed oder partial deklariert werden

```
public abstract record Person(string Name, int Age);

public record Teacher(string Name, int Age, string Subject) : Person(Name, Age);
```

# WITH EXPRESSION

- Die with Expression ermöglicht eine nicht-destruktive Mutation von Record Types
- Sie erstellt eine Kopie eines Record-Objekts und ermöglicht es, bestimmte Eigenschaften zu ändern, ohne das Originalobjekt zu verändern

```
internal class Program
{
    public static void Main()
    {
        Person person = new Person("Peter", 50);
        Person person2 = person with { Name = "John" };
        Console.WriteLine(person2); // Person { Name = John, Age = 50 }
    }
}

public record Person(string Name, int Age);
```

# RECORD STRUCTS

- Record Structs sind eine Erweiterung der Record Types, die mit C# 10 eingeführt wurden
- Sie kombinieren die Vorteile von Records, die auf Wertbasis verglichen werden, mit der Effizienz von Structs, die als Werttypen fungieren
- Im Gegensatz zu Records, die als Referenztypen fungieren, sind Record Structs Werttypen

```
public record struct Person(string Name, int Age);
```

- Record Structs sind standardmäßig veränderbar
- Um ein Record Struct unveränderlich zu machen, muss das Schlüsselwort readonly verwendet werden

```
public readonly record struct Person(string Name, int Age);
```