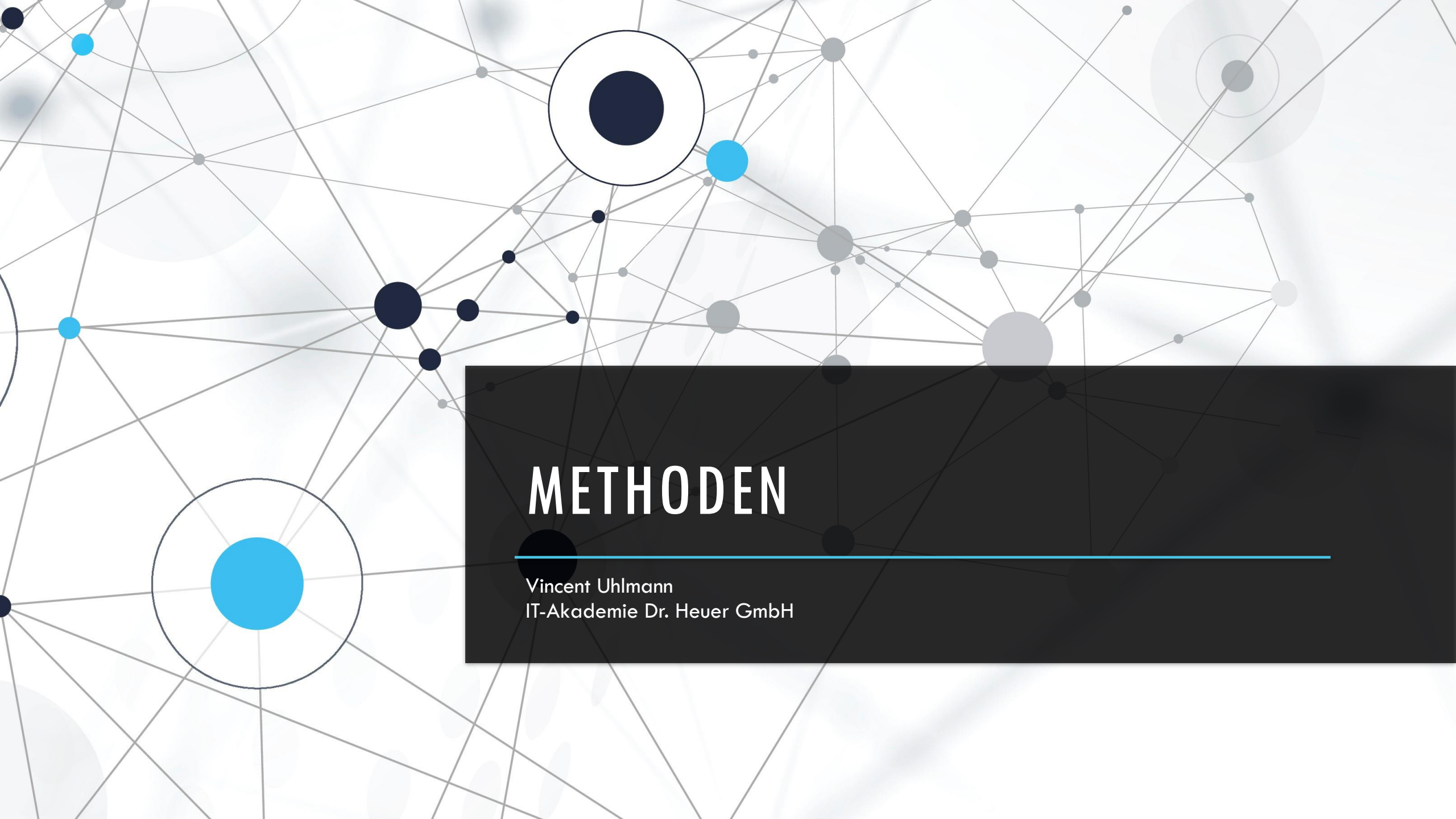

METHODEN

Vincent Uhlmann
IT-Akademie Dr. Heuer GmbH

WOZU BRAUCHEN WIR METHODEN

- Strukturierte Programmierung fördert die klare Gliederung von Code
- Durch die Zerlegung in kleinere, leicht verständliche Teile wird der Code übersichtlicher und leichter zu warten
- Funktionen können an verschiedenen Stellen im Code wiederverwendet werden

AUFRUF VON METHODEN

- Methoden innerhalb von Klassen definieren
- Reihenfolge der Funktionen in der Klasse ist beliebig
- Rückgabetyp und Methodenname charakterisieren die Methode
- Die Parameterliste definiert die benötigten Eingabewerte für die Funktion
- Funktionsnamen beginnen in .NET immer mit einem Großbuchstaben
- Für Methoden die nichts zurückgeben, wird der Rückgabetyp void verwendet

METHODEN

- Eine Methode ist eine benannte Codeeinheit, die eine bestimmte Aufgabe ausführt
- Die Syntax lautet: <Zugriffsmodifizierer><Rückgabetyp><Methodenname>(Parameterliste)
- Unsere Methoden sind alle **public static**, mehr dazu im nächsten Modul

```
public static int Add(int a, int b)
{
    return a + b;
}
```

ÜBERGABE VON PARAMETERN

- Jede Variable ist lokal für den Block, in dem sie definiert wurde
- Eine Funktion bildet einen eigenen Block mit einem eigenen Bereich für lokale Variablen

```
// Beispiel: Funktion mit lokaler Variable
// Variable zahl ist nur in ExampleMethod sichtbar
public static void ExampleMethod()
{
    int zahl = 10;
    Console.WriteLine(zahl);
}
```

PARAMETERLISTEN

- Parameter einer Funktion werden in den runden Klammern hinter dem Namen definiert
- Es handelt sich um Paare aus Datentyp und Name, durch Kommata getrennt
- Im Normalfall wird immer eine Kopie übergeben

```
// Beispiel: Funktion mit Parameterliste
public static void DisplayInfo(int alter)
{
    Console.WriteLine($"Alter: {alter} Jahre");
}
```

VERWENDUNG VON PARAMS IN FUNKTIONEN

- Das params-Schlüsselwort ermöglicht eine variable Anzahl von Parametern in einer Methode
- Dadurch können Funktionen mit unterschiedlichen Mengen von Argumenten aufgerufen werden

```
public static int Sum(params int[] numbers)
{
    int sum = 0;
    foreach (int number in numbers)
    {
        sum += number;
    }
    return sum;
}

int result1 = Sum(1, 2, 3);
```

REF UND OUT

- In C# wird standardmäßig die Wertübergabe verwendet
- Die Referenzübergabe eines Parameters wird durch die Präfixe ref und out ermöglicht
- Um einen ref oder out-Parameter zu verwenden, müssen sowohl die Funktionsdefinition als auch die aufrufende Funktion explizit das ref oder out-Schlüsselwort verwenden
- Alle Änderungen am Methodenparameter werden auf der originalen Variable vorgenommen

```
public static void Multiply(ref int zahl)
{
    zahl *= 2;
}
```

ÜBERLADEN VON FUNKTIONEN

- Überladen ermöglicht Funktionen mit gleichem Namen, aber unterschiedlichen Parametern
- Möglich durch unterschiedliche Signaturen (Name, Typ und Art aller Parameter von links nach rechts)

```
public static int Add(int a, int b)
{
    return a + b;
}

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

STANDARD PARAMETER WERTE

- Parameter können mit einem Standardwert versehen werden
- Parameter mit Standardwerten müssen immer am Ende der Parameterliste stehen

```
static void Main(string[] args)
{
    AddNumber(1, 2);
    AddNumber(1, 2, 3);
}

public static int AddNumber(int a, int b, int c = 0)
{
    return a + b + c;
}
```