C# Essentials

Methodes

Sander De Puydt

DE HOGESCHOOL
MET HET NETWERK

Hogeschool PXL – Elfde-Liniestraat 24 – B-3500 Hasselt
www.pxl.be - www.pxl.be/facebook

Korte inhoud

- Anatomie
- By value
- By reference
- Out
- Overloading

Anatomie

- Een methode ziet er als volgt uit:

```
// Voorbeeld 1
private static void EersteMethode()
{
    Console.WriteLine(1);
}

// Voorbeeld 2
private int TweedeMethode()
{
    return 1;
}
```

Anatomie

- Een methode ziet er als volgt uit:
  - toegang specificatie (private, public)
  - static vs non static
  - return type
  - naam van de methode
  - parameters
  - method body

Anatomie

- Een methode ziet er als volgt uit:
  - toegang specificatie: private, public
    - Toegang bepaalt van waar de methode opgeroepen kan worden

```
// Voorbeeld 1
private static void EersteMethode()
{
    Console.WriteLine(1);
}
```

Anatomie

- Een methode ziet er als volgt uit:
  - static vs non static

```
// Voorbeeld 1
private static void EersteMethode()
{
    Console.WriteLine(1);
}

// Voorbeeld 2
private int TweedeMethode(int a)
{
    return a + 1;
}
```

Anatomie

- Een methode ziet er als volgt uit:
  - static vs non static
    - statische methodes kunnen opgeroepen worden zonder object van een klasse
    - statische methodes kunnen enkel andere statische methodes oproepen

```
// Voorbeeld 1
private static void EersteMethode()
{
    Console.WriteLine(1);
}
```

Anatomie

- Een methode ziet er als volgt uit:
  - return type: bepaalt wat de methode teruggeeft

```
// Voorbeeld 1
private static void EersteMethode()
{
    Console.WriteLine(1);
}

// Voorbeeld 2
private int TweedeMethode(int a)
{
    return a + 1;
}
```

Anatomie

- Een methode ziet er als volgt uit:
  - naam van de methode

```
// Voorbeeld 1
private static void EersteMethode()
{
    Console.WriteLine(1);
}

// Voorbeeld 2
private int TweedeMethode(int a)
{
    return a + 1;
}
```

Anatomie

- Een methode ziet er als volgt uit:
  - parameters: de argumenten die meegegeven worden aan de methode

```
// Voorbeeld 1
private static void EersteMethode()
{
    Console.WriteLine(1);
}

// Voorbeeld 2
private int TweedeMethode(int a)
{
    return a + 1;
}
```

Anatomie

- Een methode ziet er als volgt uit:
  - method body

```
// Voorbeeld 1
private static void EersteMethode()
{
    Console.WriteLine(1);
}

// Voorbeeld 2
private int TweedeMethode(int a)
{
    return a + 1;
}
```

By Value

- Enkel de waarde wordt doorgegeven

```
static void Main(string[] args)
{
     int getal = 5;

     TweedeMethode(getal);
     Console.WriteLine($"waarde van integer na methode = {getal}");
}

private static void TweedeMethode(int a)
{
     a = a + 1;
     Console.WriteLine($"waarde van integer binnen methode = {a}");
}
```

By Reference

- De geheugenplaats wordt doorgegeven

```
static void Main(string[] args)
{
     int getal = 5;

     TweedeMethode(ref getal);
     Console.WriteLine($"waarde van integer na methode = {getal}");
}

private static void TweedeMethode(ref int a)
{
     a = a + 1;
     Console.WriteLine($"waarde van integer binnen methode = {a}");
}
```

Out

- “out” verwijst naar een uitgaande parameter. De methode garandeert initialisatie van de “ou” parameters

```
static void Main(string[] args)
{
     int getal = 5;

     DerdeMethode(getal, out int c);
     Console.WriteLine(c);
}
private static void DerdeMethode(int a, out int b)
{
     b = a * 10;
     Console.WriteLine("In deze methode wordt a*10 " +
     "opgeslagen in b.");
}
```

Overloading

- Methodes kunnen dezelfde naam hebben, indien ze **andere parameters gebruiken**.

```
static void Main(string[] args)
{
      int w = 1, x = 5, y = 10, z = 20;
      Console.WriteLine(Plus(w,x));
      Console.WriteLine(Plus(w,x,y));
      Console.WriteLine(Plus(w,x,y,z));
}
private static int Plus(int a, int b)
{
      return a + b;
}
private static int Plus(int a, int b, int c)
{
      return a + b + c;
}
private static int Plus(int a, int b, int c, int d)
{
      return a + b + c + d;
}
```