C# Essentials

Methodes en parameters

Koen Bloemen

Wim Bervoets
Kristof Palmaers

DE HOGESCHOOL MET HET NETWERK

Elfde-Liniestraat 24, 3500 Hasselt, www.pxl.be

- Anatomie
- By Value/By reference
- Parameters

Methodes

- Void procedures:
  - Geven geen waarde terug.
  - Vb. Console.WriteLine(), Close(),...
- Functie procedures:
  - Geven 1 waarde terug van een bepaald datatype.
  - Vb. Round(), int.Parse(), ToString(),...
- Event procedures:
  - Click(), KeyDown(), MouseEnter(),...

Methodes

- Binnen object-georiënteerd programmeren heet een procedure een methode.
- Waarom methodes maken?
  - Om herhaling van statements (code) te voorkomen.
  - De code die in methods staat is herbruikbaar (bijv. libraries).
  - Programma’s zijn gemakkelijker te lezen en korter.
  - Programma’s zijn gemakkelijker te maken (in groep).
  - Als je methodes public maakt in plaats van private:
    - kan je deze ook in andere files oproepen
    - of in andere projecten gebruiken.
- Wanneer je code dupliceert (copy-paste), probeer dit dan te veralgemenen naar een herbruikbare method.

Anatomie

- Een methode ziet er als volgt uit:

```
// Void method
private static void EersteMethode()
{
  Console.WriteLine(1);
}
// Method met return value
public int TweedeMethode(int a)
{
  return a + 1;
}
```

Anatomie

- toegang specificatie (private, public)
- static vs non static
- return type
- naam van de methode
- parameters
- method body

```
// Void method
private static void EersteMethode()
{
  Console.WriteLine(1);
}
// Method met return value
public int TweedeMethode(int a)
{
  return a + 1;
}
```

Toegang specificatie (private, public)

- Toegang bepaalt van waar de methode opgeroepen kan worden

```
// Void method
private static void EersteMethode()
{
  Console.WriteLine(1);
}
// Method met return value
public int TweedeMethode(int a)
{
  return a + 1;
}
```

Static vs non static

- statische methodes kunnen opgeroepen worden zonder object van een klasse
- statische methodes kunnen enkel andere statische methodes oproepen

```
// Void method
private static void EersteMethode()
{
  Console.WriteLine(1);
}
// Method met return value
public int TweedeMethode(int a)
{
  return a + 1;
}
```

Return type

- Bepaalt wat de methode teruggeeft
  - void geeft niets terug

```
// Void method
private static void EersteMethode()
{
  Console.WriteLine(1);
}
// Method met return value
public int TweedeMethode(int a)
{
  return a + 1;
}
```

Naam van de methode

- Elke methode heeft een naam
- Nodig om de methode te kunnen oproepen/gebruiken

```
// Void method
private static void EersteMethode()
{
  Console.WriteLine(1);
}
// Method met return value
public int TweedeMethode(int a)
{
  return a + 1;
}
```

Method parameters

- Parameters zijn argumenten die meegegeven worden aan de methode

```
// Void method
private static void EersteMethode()
{
  Console.WriteLine(1);
}
// Method met return value
public int TweedeMethode(int a)
{
  return a + 1;
}
```

Method body

- De inhoud van de methode

```
// Void method
private static void EersteMethode()
{
   Console.WriteLine(1);
}
// Method met return value
public 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}");
}
```

```
waarde van integer binnen methode = 6
waarde van integer na methode = 5
Press any key to continue . . .
```

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}");
}
```

```
waarde van integer binnen methode = 6
waarde van integer na methode = 6
Press any key to continue . . .
```

Out parameter

- “out” verwijst naar een uitgaande parameter.
- De methode garandeert initialisatie van de “out” 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.");
}
```

```
In deze methode wordt a*10 opgeslagen in b.
50
Press any key to continue . . .
```

Verschil ref vs out

- ref
  - Variabele die je doorgeeft aan de methode is buiten de methode al geïnitialiseerd!

```
int getal = 5;
TweedeMethode(ref getal);
```

  - De methode kan de waarde van de variabele lezen en aanpassen.
- out
  - Variabele die je doorgeeft aan de methode is buiten de methode MOET NIET geïnitialiseerd zijn!

```
int getal = 5;
DerdeMethode(out int c);
```

  - De methode moet zelf een waarde toekennen.

```
public static void DerdeMethode(out int getal)
{
  getal = 10; // BELANGRIJK: INITIALISEREN BINNEN DE METHOD!!!
  getal += 10;
}
```

Method overloading

- Methodes doen exact hetzelfde, maar parameters hebben ander datatype

```
private static int PlusMethodInt(int a, int b)
{
  return a + b;
}
private static int PlusMethodDouble(double a, double b)
{
  return a + b;
}
```

Method overloading

- Beide functies kunnen dezelfde naam gebruiken

```
private static int PlusMethod(int a, int b)
{
  return a + b;
}
private static int PlusMethod(double a, double b)
{
  return a + b;
} double
```

- Afhankelijk doorgegeven parameters wordt juiste method opgeroepen

```
int myNum1 = PlusMethod(8, 5);
double myNum2 = PlusMethod(4.3, 6.26);
Console.WriteLine("Int: " + myNum1);
Console.WriteLine("Double: " + myNum2);
```

Method 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;
}
```