leren. durven. doen.

SYNTRA WEST

Leren Programmeren

METHODEN

west-vlaanderen POM

Inhoud

1. Gebruik van methoden
   - Wat is een methode?
   - Waarom methoden gebruiken?
   - Declaratie en Creatie van een methode
   - Aanroepen van Methode
- Methode met Parameters
   - Parameters doorgeven
   - Teruggeven van waarden
- Beste Praktijken

Wat is een methode?

- Een methode is een soort codeblok dat een klein probleem implementeert
  - Een stuk code dat een naam heeft en kan aangeroepen worden vanuit andere code
  - Kan parameters aannemen en een return-waarde teruggeven
- Methods laten programmeurs toe om grote programma's op te bouwen met eenvoudiger blokken van code
- Methods worden ook gerefereerd als functions, procedures of subroutines

Waarom methoden gebruiken?

- Meer beheerbare programmeercode

  Door Opsplitsen van grote problemen in kleinere stukken

  - Betere organisatie van het programma
  - Verbeteren van leesbaarheid en verstaanbaarheid code
  - Verbeteren van onderhoud code
  - Code herbruiken door meerdere keren dezelde methoden aan te roepen

Declaratie en Creatie van methode

Declaratie en Creatie van methode

```
static void PrintLogo()
{
    Console.WriteLine("Microsoft");
    Console.WriteLine("www.microsoft.com");
}
```

Naam van methode

- Elke methode heeft een **naam**
  - Wordt gebruikt om de method aan te roepen
  - Beschrijft het doel van de methode

Declaratie en Creatie van methode (2)

```
static void PrintLogo()
{
    Console.WriteLine("Microsoft");
    Console.WriteLine("www.microsoft.com");
}
```

- **static** methods kunnen door andere methods worden aangeroepen (static or not)
  - Wordt later meer in detail besproken
- Het keyword **void** betekent dat de method geen resultaat teruggeeft

Declaratie en Creatie van methode (3)

```
static void PrintLogo()
{
    Console.WriteLine("Microsoft");
    Console.WriteLine("www.microsoft.com");
}
```

Methode body

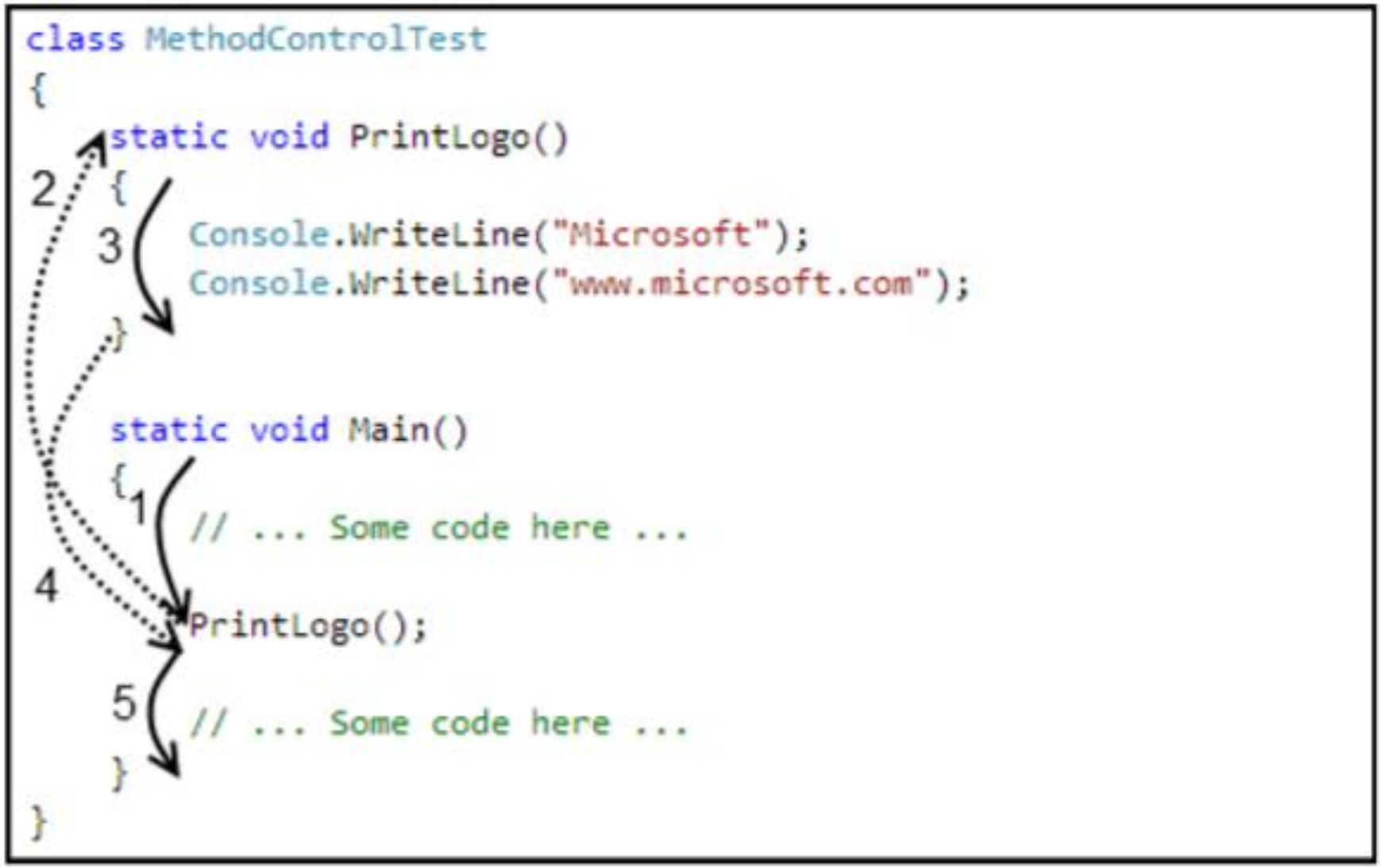
- Elke methode heeft een **body**
  - Deze bevat de programmeercode
  - Wordt afgebakend met **{** en **}**

Declaratie en Creatie van methode (4)

```
public class Program
{
    public static void Main()
    {
        PrintLogo();
}

    static void PrintLogo()
    {
        Console.WriteLine("Microsoft");
        Console.WriteLine("www.microsoft.com");
    }
}
```

- Methode worden gedeclareerd in een class
- Main() is ook een method

Declaratie en Creatie van methode (5)



Aanroepen van Methoden

Aanroepen van Methode

- Om methode aan te roepen, gebruik:
  1. Naam van method
  2. Haakjes **()**
  3. Een puntkomma (**;**)

```
PrintLogo();
```

- Hierdoor wordt de code van de methode uitgevoerd en zal het volgende op de console komen:

```
Microsoft
www.microsoft.com
```

Oproepen van Methods (2)

- Een method kan aangeroepen worden vanuit:
  - de `Main()` methode

```
public static void Main()
{
    //...
    PrintLogo();
    //....
}
```

  - Een andere methode
  - Zichzelf (recursie)

Declaratie en aanroepen van methoden

Live Demo

Oefeningen Declareren en aanroepen van Methode

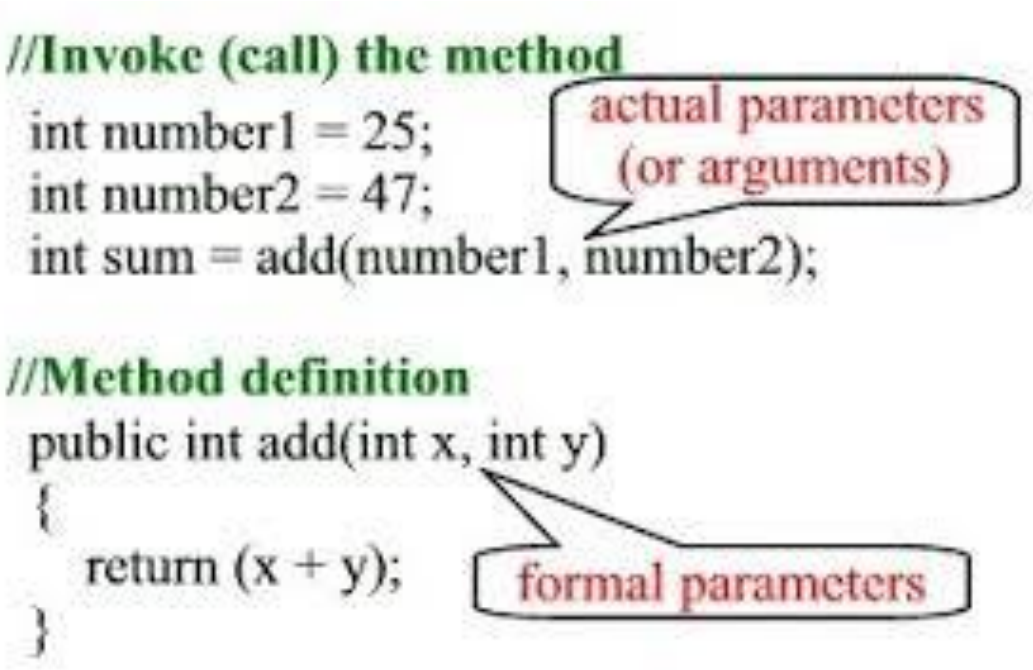
Maak de volgende 2 oefeningen op w3schools:
https://www.w3schools.com/cs/exercise.asp?filename=exercise_methods1
https://www.w3schools.com/cs/exercise.asp?filename=exercise_methods2

Schrijf een methode die de gebruiker zijn naam vraagt en die “Hallo, <naam>” op de console weergeeft.

Roep de methode aan vanuit de Main() methode.

Methoden met Parameters

Meegeven van Parameters en waarden teruggeven



Methode Parameters

- Om informatie door te geven aan een methode, kunnen parameters worden gebruikt in de declarative van de methode
  - Je kan geen, 1 of meer input waarden meegeven
  - Je kan waarden van verschillende data types doorgeven
  - Elke parameter heeft een **naam** en **data type**
  - Aan de parameters worden bepaalde waarden toegewezen wanneer de method wordt aangeroepen

Definiëren en gebruiken van method Parameters

```
static void PrintSign(int number)
{
    if (number > 0)
        Console.WriteLine("Positief");
    else if (number < 0)
        Console.WriteLine("Negatief");
    else
        Console.WriteLine("Nul");
}
```

1 parameter

- Het gedrag van de method is afhankelijk van parameter
- Parameters kunnen van elk type zijn
  - `int`, `double`, `string`, `DateTime` …

Declaratie en gebruik van Method Parameters (2)

- Methods kunnen meerdere parameters bezitten:

```
static void PrintMax(float number1, float number2)
{
    float max = number1;
    if (number2 > number1)
    {
        max = number2;
    }
    Console.WriteLine("Maximal number: {0}", max);
}
```

2 parameters

- De volgende syntax is **niet** geldig:

```
static void PrintMax(float number1, number2)
```

Aanroepen van methods met Parameters

- Om method aan te roepen en waarden aan parameters toe te kennen:
  - Gebruik de naam van method, gevolgd door de waarden voor de parameters (deze worden *argumenten* genoemd)
- Voorbeelden:

```
PrintSign(123);
PrintSign(-5);
PrintSign(2 + 3);
PrintMax(2.3f, 6.8f);
PrintMax(100, 200);
PrintMax(oldQuantity * 1.5, quantity * 2);
```

Aanroepen van Methoden met Parameters (2)

- Expressies moeten van hetzelfde type zijn als the parameters van de method (of compatible type)
  - Bv Indien de method een `float` expressie verwacht, kan je bv ook `int` doorgeven
- Gebruik **dezelfde volgorde** als parameters in de declaratie van de method
- Ook voor methods zonder parameters moeten de haakjes worden gezet bij de aanroep: **()**

Methode Parameters – Voorbeeld

```
static void PrintSign(int number)
{
    if (number > 0)
        Console.WriteLine("Het getal {0} is positief.", number);
    else if (number < 0)
        Console.WriteLine("Het getal {0} is negatief.", number);
    else
        Console.WriteLine("Het getal {0} is nul.", number);
}
static void PrintMax(float number1, float number2)
{
    float max = number1;
    if (number2 > number1)
    {
        max = number2;
    }
    Console.WriteLine("Grootste getal: {0}", max);
}
```

Aanroepen van Methoden met Parameters

Voorbeelden

Verschil tussen Parameters en Argumenten van een methode

- **Parameters** van method worden gedefinieerd bij declaratie van method

```
static void PrintMax(float number1, float number2)
```

- **Argumenten** van method: worden gespecifieerd bij aanroep van de method

```
PrintMax(100.0f, -23.5f);
```

Methode Parameters en argumenten

Live Demo

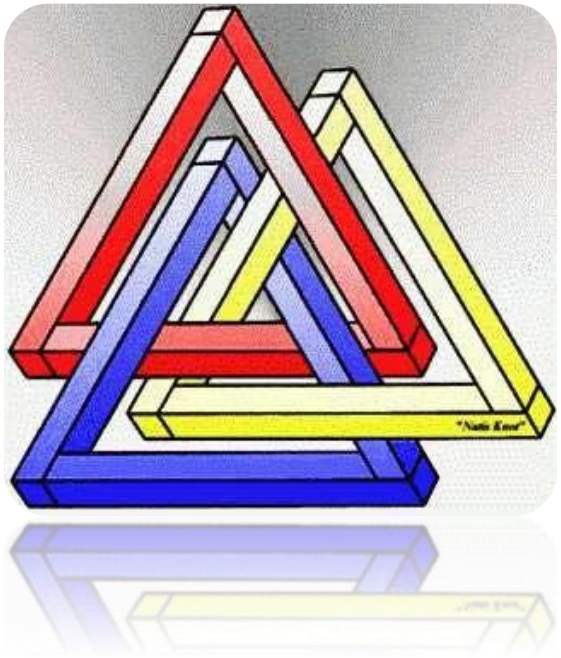
Oefening methode met parameters

1. Maak de volgende oefening op w3schools:
https://www.w3schools.com/cs/exercise.asp?filename=exercise_methods3

2. Schrijf een methode die de gebruiker zijn naam vraagt en die “Hallo, <naam>” op de console weergeeft.

Gebruik in de declaratie van de methode één parameter naam van het type string

Roep deze methode aan vanuit de Main() methode.



Printen van driehoek

Live Demo

Printen van driehoek – Voorbeeld

- Creëren van een programma om driehoeken af te printen op de volgende manier:

n=5 →

```
1
1 2
1 2 3
1 2 3 4
1 2 3 4 5
1 2 3 4
1 2 3
1 2
1
```

n=6 →

```
1
1 2
1 2 3
1 2 3 4
1 2 3 4 5
1 2 3 4 5 6
1 2 3 4 5
1 2 3 4
1 2 3
1 2
1
```

Printen van driehoek – Voorbeeld

```
public static void Main()
{
    Console.WriteLine("Geef een geheel getal > 0:");
    int getal = int.Parse(Console.ReadLine());
    for (int line = 1; line <= getal; line++)
    {
        PrintLine(1, line);
    }
    for (int line = getal - 1; line >= 1; line--)
    {
        PrintLine(1, line);
    }
}
static void PrintLine(int start, int end)
{
    for (int i = start; i <= end; i++)
    {
        Console.Write(" {0}", i);
    }
    Console.WriteLine();
}
```

Optionele Parameters

- Vanaf C# 4.0 zijn optionele parameters with mogelijk en default waarden toegekend bij de declaratie:

```
static void PrintNumbers(int start = 0, int end = 100)
{
  for (int i = start; i <= end; i++)
  {
    Console.Write("{0} ", i);
  }
}
```

- De bovenstaande methode kan aangeroepen worden op verschillende manieren:

```
PrintNumbers(5, 10);
PrintNumbers(15);
PrintNumbers();
PrintNumbers(end: 40, start: 35);
```

Optionele Parameters

Live Demo

Oefening methode met Optionele Parameters

- Maak een methode met naam **BegroetPersoon** met twee optionele parameters. De eerste optionele parameter heeft de naam **geslacht** en als standaard waarde “” (lege string), de 2de parameter is **uur** van de dag (geheel getal) en heeft als standaard waarde 0
- Test de volgende manieren van aanroepen van de methode uit:

```
BegroetPersoon("vrouw", 10);          //Goede morgen Mevrouw
BegroetPersoon(uur:15);               //Goede namiddag
BegroetPersoon();                     //Goedendag
BegroetPersoon("man");                //Goedendag Mijnheer
BegroetPersoon(uur:17);               //Goede avond
BegroetPersoon(uur: 12, geslacht: "man"); //Goede middag Mijnheer
```

```
Goede morgen Mevrouw
Goede namiddag
Goedendag
Goedendag  Mijnheer
Goede avond
Goede middag Mijnheer
```

Waarde teruggeven vanuit Methode

Waarde teruggeven vanuit Methode

- Een method kan een waarde teruggeven (return) naar zijn oproeper
- Teruggegeven waarde (return value):
  - Kan toegewezen worden aan een variable, bv:

```
string message = Console.ReadLine();
// Console.ReadLine() returns a string
```

  - Kan gebruikt worden in expressies; bv:

```
float price = GetPrice() * quantity * 1.20;
```

  - Kan doorgegeven worden aan een andere methode:

```
int age = int.Parse(Console.ReadLine());
```

Declaratie van Methode die een waarde teruggeeft

- In plaats van `void`, specifieer het data type van de waarde die wordt terug gegeven, bv `int` :

type van return value

```
static int Multiply(int firstNum, int secondNum)
{
    return firstNum * secondNum;
}
```

- Methods kunnen elk soort data type (`int`, `string`, array, ...) teruggeven
- `void` methods geven niets terug (hebben geen return value)
- De ***combinatie van de naam van de method, zijn parameters, return value*** wordt ***methode signatuur*** genoemd
- Gebruik `return` keyword om de waarde terug te geven

`return` Statement

- `return` statement:
  - Zal onmiddellijk de uitvoering van method stoppen
  - Geeft een waarde terug aan de oproeper
  - Bijvoorbeeld:

```
return -1;
```

- Om een method met `void` return waarde te beëindigen, gebruik:

```
return;
```

- return kan meermaals worden gebruikt in een methode-body

Voorbeeld terugkeerwaarde (return value) methode

Omzetting temperatuur – Voorbeeld

- Zet de temperatuur om van Fahrenheit naar graden Celsius:

```
static double FahrenheitToCelsius(double degrees)
{
    double celsius = (degrees - 32) * 5 / 9;
    return celsius;
}
public static void Main()
{
    Console.Write("Temperatuur in Fahrenheit: ");
    double t = Double.Parse(Console.ReadLine());
    t = FahrenheitToCelsius(t);
    Console.Write("Temperatuur in graden Celsius: {0}", t);
}
```

Oefeningen Return waarde methode

- Maak de volgende oefening op w3schools:

https://www.w3schools.com/cs/exercise.asp?filename=exercise_methods4

- Schrijf een methode ‘GetMax’ met 2 int parameters. De methode returnt het grootste. Schrijf een programma dat 3 integers inleest van de console en het grootste toont, gebruikmakend van de GetMax methode.

- Schrijf een methode dat het laatste cijfer van een meegegeven positief getal omzet in een string. Bv: 512 ➔ “twee”, 25 ➔ “vijf”

Waarde teruggeven vanuit Methode

Voorbeeld input validatie

Input Validatie – Voorbeeld

- Validatie van input data:

```
public static void Main()
{
    Console.WriteLine("Hoe laat is het?");
    Console.Write("Uren: ");
    int hours = int.Parse(Console.ReadLine());
    Console.Write("Minuten: ");
    int minutes = int.Parse(Console.ReadLine());
    bool isValidTime = ValidateHours(hours) && ValidateMinutes(minutes);
    if (isValidTime)
    {
        Console.WriteLine("Het is {0}:{1}", hours, minutes);
    }
    else
    {
        Console.WriteLine("Ongeldige tijd!");
    }
}
        // (vervolg voorbeeld op volgende slide…)
```

Input Validatie – vervolg voorbeeld

```
static bool ValidateMinutes(int minutes)
{
    bool result = (minutes >= 0) && (minutes < 60);
    return result;
}

static bool ValidateHours(int hours)
{
    bool result = (hours >= 0) && (hours < 24);
    return result;
}
```

Input Validatie - Voorbeeld

Live Demo

‘Overloading’ Methoden

Meerdere Methoden met dezelfde naam

‘Overloading’ Methoden

- Wat betekent: "overload een methode"?

  = Gebruik van dezelfde naam van een methode voor andere methode met andere parameters, bv:

```
static void Print(string text)
{
    Console.WriteLine(text);
}
static void Print(int number)
{
    Console.WriteLine(number);
}
static void Print(string text, int number)
{
    Console.WriteLine(text + ' ' + number);
}
```

Method Overloading

Live Demo

Oefening method overloading

- **Overload de methode met naam Add, zodat deze de som maakt :**

- **Van 3 gehele getallen**
- **Van 3 komma getallen**

.

Maak gebruik van de volgende code om Add aan te roepen:

```
21      public static void Main()
22      {
23
24          int sum2 = Add(1, 2, 3);
25          Console.WriteLine("som van de 3  "
26                            + "gehele waarden : " + sum2);
27          double sum3 = Add(1.0, 2.04, 3.5);
28          Console.WriteLine("som van de 3 "
29                            + "double waarden: {0:F2} " , sum3);
30      }
```

Variabel aantal Parameters

Variabel aantal Parameters

- Een method in C# kan een variabele een verschillend aantal parameters aanvaarden bij het gebruik van keyword `params` keyword, vb:

```
static long CalcProduct(params int[] elements)
{
    long product = 1;
    foreach (int element in elements)
    {
        product *= element;
    }
    return product;
}
public static void Main()
{
    Console.WriteLine(CalcProduct(2, 5)); //10
    Console.WriteLine(CalcProduct(4, 1, -2, 12)); //-96
    Console.WriteLine(CalcProduct(5));//5
}
```

Oefening params keyword

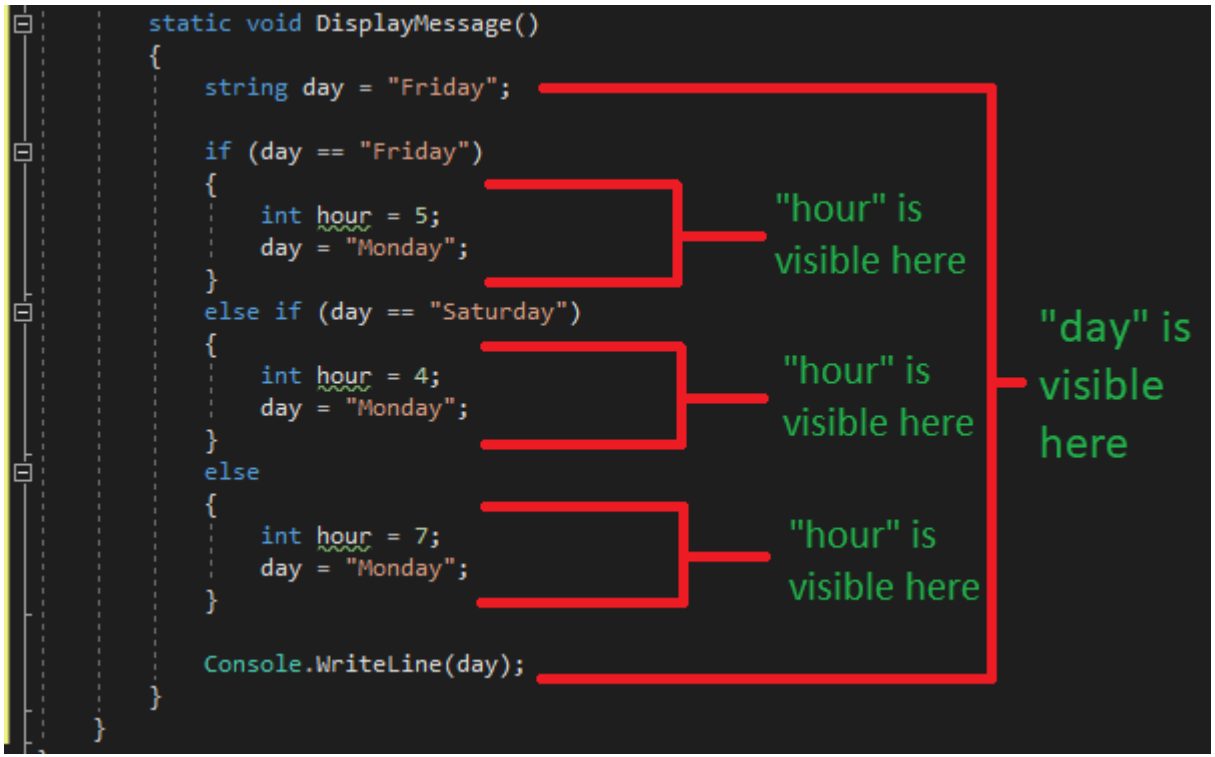
- Schrijf 1 methode Som die ervoor zorgt dat het volgende programma werkt.
  Je mag niets aanpassen aan de Main methode

```
public static void Main()
{
    int getal1 = 5;
    int getal2 = 3;
    int getal3 = 6;

    int som = Som(getal1, getal2, getal3);
    Console.WriteLine(som); //geeft 14

    som = Som(getal1, getal2);
    Console.WriteLine(som); //geeft 8

    som = Som(getal1);
    Console.WriteLine(som); //geeft 5

    som = Som();
    Console.WriteLine(som); //geeft 0
}
```

Scope van variabelen

Scope van een variabele: is het ‘bereik’ van een variabele, dwz. Waar ze in de code worden herkend.

De Scope van een lokale variabele is binnen de body (dus de binnen de {}-tekens waartussen ze gedeclareerd is) waarin ze zijn gedeclareerd, bv:



‘Scope’ van variabelen - methoden

Variabelen gedeclareerd in een methode zijn enkel gekend binnen deze methode

```
void Functie1()
{
    int i_test = 33;
}
void Functie2()
{
    int i_test = 44;
}
```

Beide i_test variabelen weten niet van elkaar dat ze bestaan.

Methoden – Beste praktijken

- Elke method zou (bij voorkeur) **één enkele, welgedefinieerde taak moeten uitvoeren**
- De **naam** van de method beschrijft de taak op een **duidelijke, ondubbelzinnige** manier
  - Goede voorbeelden: `CalculatePrice`, `ReadName`
  - Slechte voorbeelden: `f`, `g1`, `Process`
  - In C# methoden zouden moeten beginnen met een hoofdletter **(naamgevingsconventie)**
- Vermijd methods langer dan een scherm
  - Splits ze op tot meerdere **kortere** methods

Samenvatting Methoden

- Opdelen van grote programma's in eenvoudige methoden die kleine sub-problemen oplossen
- methoden bestaan uit de declaratie en body
- methoden worden via hun naam aangeroepen
- methoden kunnen parameters aanvaarden
  - Parameters nemen de waarden aan bij het aanroepen van de methode
- methoden kunnen een waarde teruggeven van een bepaald data type of helemaal niets (void)

Methoden

leren. durven. doen.

SYNTRA WEST

Vragen?

west-vlaanderen POM

Referenties

http://learncs.org/
https://www.w3schools.com/cs
https://dotnetfiddle.net/
https://www.rusoaica.com/functions/variable-scope/

Fundamentals of computer programming with c#