## Lesbrief 6 Methoden en parameters

| * Probeer de voorbeelden uit. * Voer de opdrachten uit. |
| --- |

Als je programma steeds meer regels code bevat is het handig om deze voor de **leesbaarheid** op te delen in methoden. In principe deel je een probleem op in kleinere deelproblemen, waarbij elk deelprobleem 1 of meerdere methoden omvat.

Een methode heeft ook als doel **hergebruik**. Als je in jouw programma bijvoorbeeld regelmatig een berekening moet uitvoeren dan hoef je dit maar 1x te programmeren.

Methoden zijn onderdeel van een **Class** of een **Struct**.

De syntax ziet er als volgt uit:

*<access level> <optioneel modifiers> <return value> <name> (<parameters>)*

* De access level wordt in een volgend lesbrief behandeld. Voor nu hanteren we “public” als access level.
* Modifiers zijn optioneel en voegen bepaalde eigenschappen toe aan een methode, zoals een eigenschap dat een methode maar 1 x kan bestaan, ook wel static genoemd.
* Return value geeft aan of de methode een waarde retourneert. Zo ja dan staat hier het datatype van de waarde die geretourneerd wordt. Anders staat er “void”.
* Name is de naam van de methode. Deze beginnen met een **hoofdletter**.
* Parameters is de lijst van variabelen die aan de methode wordt doorgegeven.

**Voorbeelden**:

public static void Start()

public static string Concat(string firstName, string lastName)

public static int GetMax(int a, int b)

**Voorbeeldprogramma**

Onderstaand programma print een header, vervolgens “Hello World” en dan nogmaals de header op het scherm.

| using System;  namespace HelloWorld {  class Program  {  static void Main(string[] args)  {  PrintHeader();  Console.WriteLine("Hello World");  PrintHeader();  Console.ReadLine();  }   public static void PrintHeader()  {  Console.WriteLine("\*\*\*\*\*\*\*\*\*");  Console.WriteLine("Message from JUPITER");  Console.WriteLine("\*\*\*\*\*\*\*\*\*");  }  } } |
| --- |

De methode “PrintHeader” heeft de volgende eigenschappen:

| public static void PrintHeader() |
| --- |

* public = access level
* static = modifier
* void = return value
* PrintHeader = naam van de methode
* Geen parameters.

De methode “PrintHeader” krijgt geen parameters mee. Ook geeft de methode geen waarde terug aan de aanroeper. Daarom wordt als return waarde “**void**” gehanteerd.

De methode “PrintHeader” wordt 2 x aangeroepen (hergebruik).

| PrintHeader();  Console.WriteLine("Hello World");  PrintHeader(); |
| --- |

**Voorbeeldprogramma met parameter**

In onderstaand programma gebruiken we dezelfde methode om een header en footer te printen, we hergebruiken de methode “PrintLabel”.

Aangezien alleen de tekst verschilt wordt deze als **parameter** meegegeven.

| using System;  namespace HelloWorld {  class Program  {  static void Main(string[] args)  {  PrintLabel("Message from JUPITER");  Console.WriteLine("Hello World");  PrintLabel("End of message");  Console.ReadLine();  }   public static void PrintLabel(string message)  {  Console.WriteLine("\*\*\*\*\*\*\*\*\*");  Console.WriteLine(message);  Console.WriteLine("\*\*\*\*\*\*\*\*\*");  }  } } |
| --- |

De methode “PrintLabel” heeft de volgende eigenschappen:

| public static void PrintLabel(string message) |
| --- |

* public = access level
* static = modifier
* void = return value
* PrintLabel = naam methode
* string message = parameter

Bij het definiëren van de methode “PrintLabel” definiëren we ook de parameter. Bij het definiëren van de parameter geef je de **datatype** mee gevolgd door een **naam**: “string message”.

De methode “PrintLabel” wordt 2 x aangeroepen, eerst met de tekst “Message from Jupiter” als parameter en vervolgens met de tekst “End of message” als parameter.

| PrintLabel("Message from JUPITER");  Console.WriteLine("Hello World");  PrintLabel("End of message"); |
| --- |

De methode “PrintLabel” retourneert geen waarde aan de aanroeper dus is de return value een **void**.

**Voorbeeldprogramma met parameter en return**

In onderstaand programma wordt de gebruiker gevraagd om 2 getallen. Vervolgens is er een methode “getMax” die het grootste van de 2 getallen retourneert. Beide getallen worden als parameter meegegeven.

| using System;  namespace GrootsteGetal {  class Program  {  static void Main(string[] args)  {  Console.WriteLine("Voer het eerste getal in: ");  int getal1 = Convert.ToInt32(Console.ReadLine());  Console.WriteLine("Voer het tweede getal in: ");  int getal2 = Convert.ToInt32(Console.ReadLine());  int max = GetMax(getal1, getal2);  Console.WriteLine("Het grootste getal is " + Convert.ToString(max));  }   public static int GetMax(int a, int b)  {  if (a > b)  {  return a;  }  return b;  }  } } |
| --- |

In dit programma is:

| public static int GetMax(int a, int b) |
| --- |

* public = access level
* static = modifier
* int = return value
* GetMax = naam methode
* int a, int b = parameters

Aan de methode “GetMax” worden 2 parameters meegegeven, 2 getallen a en b van datatype int.

De methode “GetMax” retourneert een waarde namelijk de grootste van de 2 getallen. Om een waarde te retourneren wordt gebruik gemaak van het keyword **return**.

| return b; |
| --- |

Omdat er een waarde geretourneerd wordt geven we het datatype van de return variabele op in de methode declaratie. In bovenstaand voorbeeld is dat een **int**.

| public static int GetMax(int a, int b) |
| --- |

**Math**

In bovenstaande voorbeeld kunnen we ook gebruik maken van de **Math** klasse. Deze klasse bevat allerlei methoden om mathematische operaties uit te kunnen voeren.

Bijvoorbeeld Math.Abs(), Math.Pow(), Math.Sin() en nog veel meer. Voor meer informatie over de Math klasse, zie <https://docs.microsoft.com/en-us/dotnet/api/system.math?view=netframework-4.7.2>

De methode ziet er dan als volgt uit:

| public static int getMax(int a, int b)  {  return Math.Max(a, b);  } |
| --- |

| **Opdracht**   * Pas het Boodschappen programma aan zodanig dat de BTW in een aparte methode wordt berekend. Deze methode krijgt als parameter het aantal, de prijs exclusief BTW en het BTW percentage mee en retourneert wat het bedrag aan te betalen BTW is. |
| --- |