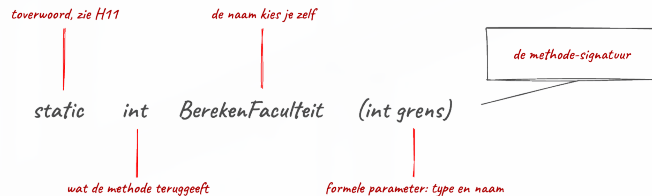




// 01

Wat is een methode?

Een blok code met een naam: **één keer** schrijven, overal aanroepen. Herkenbaar aan de **ronde haakjes**. Ook `Console.WriteLine()` en `Main` zijn methoden. De eerste lijn is de **methode-signatuur**:



```
static void ToonTitel()
{
    Console.WriteLine("Timsoft XP");
}
ToonTitel(); // aanroepen, bv. vanuit Main
```

Je methoden staan in `class Program`, naast `Main`, in eender welke volgorde.

// 02

Returntype en return

- `void`: de methode geeft niets terug
- Elk ander type: `return` geeft een waarde van dat type terug
- `return` stopt de methode meteen (in een `void`: `return;`)

```
static string VerkrijgAuteurNaam()
{
    return "Tim Dams";
}
string naam = VerkrijgAuteurNaam();
```

// LET OP

Geen `void`? Dan moet **elk** pad een `return` bereiken. Anders: *Not all code paths return a value.*

// 03

Parameters

Bij `BerekenFaculteit(getal)` is `getal` de **actuele** en `grens` de **formele parameter**. De namen hoeven niet gelijk te zijn: enkel de waarde reist mee. De **volgorde** telt. Met

`ToonInfo(string naam, int leeftijd):`

✓ `ToonInfo("Tim", 37);` ✗ `ToonInfo(37, "Tim");`

// 04

By value: een kopie

De methode werkt met een kopie. Het origineel verandert niet.

```
static void JaartjeOuder(int leeftijd)
{
    leeftijd++;
}
int mijnLeeftijd = 40;
JaartjeOuder(mijnLeeftijd);
Console.WriteLine(mijnLeeftijd);
```

UITVOER
40

// LET OP

`Verdubbel(punten);` verandert `punten` niet. Vang het resultaat op: `punten = Verdubbel(punten);`

// 05

Methoden combineren

Een methode mag andere methoden aanroepen. Ze doet **één ding**: heeft de naam een "En" nodig, splits ze dan op.

✗ Nooit een methode **in** een methode (boete: 3 punten).

// LET OP

Een methode die zichzelf aanroept zonder stopconditie crasht met een `StackOverflowException`.

// 06

Bibliotheken en IntelliSense

- `Console.` toont wat de bibliotheek kan (`Ctrl+Spatie`)
- Na het `(` zie je elke signatuur (`Ctrl+Shift+Spatie`)
- `F1` opent de online documentatie
- Jouw `///`-commentaar verschijnt mee in die popup
- Step in**: bij het debuggen stap je mee in je methode

// 07

Named parameters

Met `ToonPunten(string vak, int score, string student):`

```
ToonPunten("Steven", 14, "Programmeren");
ToonPunten(student: "Steven", score: 14,
            vak: "Programmeren");
```

UITVOER

Programmeren behaalde 14/20 voor Steven
Steven behaalde 14/20 voor Programmeren

Alles benoemd: vrije volgorde. Gemengd: elk op zijn plaats.

// 08

Optionele parameters

Met een default waarde mag je ze weglaten. Altijd **achteraan**.

```
static void ToonFactuur(string klant,
                       int korting = 0, int btw = 21)
{
    // toont een factuurlijn
}
```

AANROEP	KORTING	BTW
<code>ToonFactuur("Tim");</code>	0	21
<code>ToonFactuur("Tim", btw: 6);</code>	0	6

// LET OP

Weglaten kan enkel van achter naar voor.

`ToonFactuur("Tim", 6);` compileert, maar de 6 landt in `korting`.

// 09

Method overloading

Zelfde naam, andere parameterlijst (type en/of aantal).

```
static int Verdubbel(int getal)
{
    return getal * 2;
}
static string Verdubbel(string tekst)
{
    return tekst + tekst;
}
```

De compiler kiest: `Verdubbel(21)` geeft `42`, `Verdubbel("ha")` geeft `"haha"`.

// LET OP

Verschilt enkel het returntype? Fout: *already defines a member.*