



// 01

Klasse en object

Een **klasse** is de blauwdruk, een **object** een instantie ervan, met een eigen **toestand**, **gedrag** en **identiteit**. Een klasse is een nieuw datatype: je schrijft ze met `class` in een eigen bestand (Project > Add class).

```
internal class Auto { } // Auto.cs
Auto mijnAuto = new Auto(); // in Main
```

`new` geeft het **adres** van het nieuwe object terug.

// LET OP

Een objectvariabele zonder object is `null`. Roep je er toch een methode of property op aan, dan crasht je programma:

```
NullReferenceException.
```

// 02

Abstractie en encapsulatie

Het **black-box**-principe: je rijdt met het stuur en de pedalen (de **interface**), zonder de motor te kennen. **Encapsulatie** steekt alles wat samen hoort in één klasse, **abstractie** toont de buitenwereld enkel een eenvoudige interface. Bij `Random` volstaan `new` en `Next()`. De logica zit in het object zelf:

```
ball.Update();
```

// 03

Access modifiers

- `public`: de buitenwereld kan erbij met de **dot-operator**
- `private`: enkel zichtbaar binnen de klasse zelf
- Niets ervoor betekent `private`. Een klasse zonder modifier is `internal`: enkel bruikbaar in je eigen project.

```
Mens rachid = new Mens();
rachid.Praat(); // public: werkt
rachid.VertelGeheim(); // private: fout
```

Binnen de klasse mag `Praat` wel `VertelGeheim()` aanroepen.

// 04

Instantievariabelen

Elk object bewaart zijn eigen waarden.

```
internal class Mens
{
    private int geboorteJaar = 1970;
    public void StartVerjongingskuur()
    {
        geboorteJaar++;
    }
    public void Praat() // geen static
    {
        Console.WriteLine(geboorteJaar);
    }
}
```

```
Mens elvis = new Mens();
Mens bono = new Mens();
elvis.StartVerjongingskuur();
elvis.Praat();
bono.Praat();
```

UITVOER
1971
1970

Namen: instantievariabelen klein, de rest met een hoofdletter.

// LET OP

Een instantievariabele maak je **nooit** `public`: dan kan iemand `geboorteJaar` op `-12000` zetten. Gebruik een property.

// 05

Full properties

Gecontroleerde toegang tot een instantievariabele: `get` geeft de waarde met `return`, `set` krijgt de nieuwe waarde in `value`.

```
private int energie;
public int Energie
{
    get { return energie; }
    set
    {
        if (value >= 0)
            energie = value;
    }
}
```

```
vader.Energie = 20; // set
vader.Energie = -1; // set, maar geen effect
Console.WriteLine(vader.Energie); // get: 20
```

Naam van de instantievariabele, met een hoofdletter. VS: `propfull` en 2x Tab.

// 06

Property-variaties

SOORT	HOE	BIJVOORBEELD
write-only	geen <code>get</code>	een pincode
read-only	geen <code>set</code>	enkel uitlezen
private set	<code>private set</code>	enkel de klasse past aan
transformerend	<code>get</code> berekent	<code>VolledigeNaam</code>

Actie: **methode**. Eigenschap, snel te berekenen: **property**.

// LET OP

Ook binnen de klasse ga je via de property:

```
Energie = resetWaarde; // Wie rechtstreeks energie aanpast,
// omzeilt de controle in de set.
```

// 07

Auto-properties

Geen controle nodig? C# maakt de instantievariabele zelf. VS: `prop`, 2x Tab.

```
public string Voornaam { get; set; }
public int Geboortejaar { get; set; } = 2002;
public string Naam { get; private set; }
public string Auteur { get; } = "Tim";
```

// LET OP

In een auto-property kan je geen controle schrijven. Heb je die nodig, zet ze dan om naar een full property.

// 08

DateTime

```
DateTime nu = DateTime.Now;
DateTime trouw = new DateTime(2017, 4, 21);
DateTime einde = trouw.AddDays(35);
Console.WriteLine(einde.DayOfWeek);
```

UITVOER
Friday

- `AddDays`, `AddMonths`, ... geven een **nieuw** object terug
- Read-only properties: `Year`, `Month`, `Day`, `DayOfWeek`, ...
- Op de klasse zelf: `Now`, `Parse` en `IsLeapYear`
- Twee datums aftrekken geeft een `TimeSpan`