C# Les 5

Tuples

In les 3 gebruikten we verschillende arrays om informatie over wagens in op te slagen.

```
string[] merk = { "Volvo", "Fiat", "BMW", "Volvo", "Volvo", "BMW" };
string[] model = { "66", "Panda", "732i", "265", "340", "525" };
int[] bouwjaar = { 1975, 1981, 1979, 1980, 1982, 1974 };
int[] aantalDeuren = { 2, 3, 3, 5, 5, 4 };
string[] type = { "sedan", "hatchback", "sedan", "wagon", "hatchback", "sedan
" };
```

Maar wat als we een wagen willen verwijderen? Dan moeten we goed opletten dat we dat in elke lijst correct doen, anders stort ons kaartenhuisje in elkaar. De informatie is dus wat onhandig gegroepeerd, alle merken staan bijvoorbeeld samen terwijl het logischer zou zijn om de gegevens zo te groeperen als een rij in een tabel (denk aan Excel of SQL):

- `Volvo, 66, 1975, 2, sedan`
- `Fiat, Panda, 1981, 3, hatchback`
- …

Het zou ook handig zijn om de velden een naam te kunnen geven, bijvoorbeeld:

merk: Volvo, **model**: 66, **bouwjaar**: 1975, **aantal deuren**: 2, **type**: sedan

Daarvoor komen *tuples* goed van pas. Een tuple is een groepering van bijeenhorende informatie. Anders dan een lijst kan een tuple verschillende soorten data (= data-types) bevatten, dus je kan getallen, strings en booleans in één tuple gaan zetten. Een lijst daarentegen kan enkel informatie van hetzelfde data-type bevatten.

Een tuple maken we zo (het voorbeeld is op 2 lijnen geschreven door plaatsgebrek, het mag op 1 lijn):

```
(string merk, string model, int bouwjaar, int nDeuren, string type) auto1
= ("Volvo", "66", 1975, 2, "sedan");
```

Je kan stukjes uit de tuple gaan halen door een punt te gebruiken:

```
Console.WriteLine(auto1.merk);
```

Met een tuple kan je bijeenhorende gegevens groeperen. De gegevens in de tuple vraag je op met het punt.

Een tuple kan je, zoals elke variabele, meegeven met een methode (de methode-signatuur en Console.WriteLine zijn wegens plaatsgebrek op 2 lijnen gezet):

```
static void Main(string[] args)
{
    (string merk, string model, int bouwjaar, int nDeuren, string type) auto1
    = ("Volvo", "66", 1975, 2, "sedan");

    printAuto(auto1);
}

static void printAuto((
    string merk,
    string model,
    int bouwjaar,
    int nDeuren,
    string type) auto)
{
    Console.WriteLine($"De {auto.merk} {auto.model} is een" +
        $"{auto.nDeuren}-deurs {auto.type} en werd " +
        $"voor het eerst gebouwd in {auto.bouwjaar}.");
}
```

Een lijst van tuples

Nu hebben we 1 auto in 1 tuple gezet, maar onze winkel verkoopt meer dan 1 auto… We hebben dus een lijst van tuples nodig.

```
List<(string,string,int,int,string)> carList = new List<(
    string merk,
    string model,
    int bouwjaar,
    int nDeuren,
    string type)>
{
    ("Volvo", "66", 1975, 2, "sedan"),
    ("Fiat", "Panda", 1981, 3, "hatchback"),
};
```

Het declareren van de variabele carList (= het eerste lange stuk met `List<(string,string,int,int,string)> carList`) kan gelukkig korter gemaakt worden. C# is slim genoeg om te zien wat voor soort variabele er nodig is, omdat deze informatie al vermeld

is in het stukje achter de assignment operator (=). Het type van de variabele hoef je dus niet expliciet nog eens te vermelden. We laten C# het type impliciet vaststellen door het gebruik van het woordje *var*. Dit wordt *implicit typing* genoemd.

Dit voorbeeld maakt doet hetzelfde en is korter door gebruik te maken van implicit typing:

```
var carList = new List<(
    string merk,
    string model,
    int bouwjaar,
    int nDeuren,
    string type)>
{
    ("Volvo", "66", 1975, 2, "sedan"),
    ("Fiat", "Panda", 1981, 3, "hatchback"),

};
```

Fields

In plaats van al de gegevens door te geven aan je methodes via parameters en returns kan je bepaalde gegevens ook buiten een methode plaatsen (maar nog binnen de accolades van een class, zie in het geel hieronder). Zo zijn die gegevens vanuit al de methodes in je class bereikbaar.

```
using System;
using System.Collections.Generic;

class Program
{
    static private List<(string merk, string model, int bouwjaar, int nDeuren
, string type)> carList;

    static void Main(string[] args)
    {
        carList = new List<(
            string merk,
            string model,
```

```
            int bouwjaar,
            int nDeuren,
            string type)>
        {
            ("Volvo", "66", 1975, 2, "sedan"),
            ("Fiat", "Panda", 1981, 3, "hatchback"),

        };

        Console.WriteLine(carList[0].merk);
    }
}
```

Pas deze techniek enkel toe voor essentiële gegevens. In ons voorbeeld zetten we bijvoorbeeld de lijsten van onze auto's in een veld. Hulpvariabelen zoals tellers of variabelen voor input op te slagen worden niet in velden geplaatst omdat tot rommelige code leidt.

Je bent momenteel ook nog niet verplicht om met velden te werken.

Oefening 1: CLASSIC AUTOSHOP met tuples

A. Zet al de wagens in de carList. Maak daarna een methode printAuto**s**.

B. Voorzie een optie om auto's te verwijderen.

C. Voorzie een optie om een auto toe te voegen.

Oefening 2: Gezelschapsspellenwinkel “De Ork”

A. Maak een lijst van tuples voor onderstaande producten in een gezelschapsspellenwinkel.

Naam	Prijs	Promo	Op voorraad	Vanaf leeftijd
Catan	41,99	False	2	10
Carcassonne	27,99	True	3	8
Uno	9,99	False	5	7
Scrabble	39,99	False	3	10
Pietjesbak	14,99	True	2	8

B. Toon de gegevens van de spelen in de console. Toon voor de producten in promo de originele prijs en de promo-prijs (-20%). Kan je de promoprijs met een groene achtergrond tonen en de originele prijs met een rode achtergrond?

C. Voorzie een optie om al de producten te tonen waarvan er minder dan 3 op voorraad zijn.