C# Essentials

Generieke List en Dictionary

Lector: Tom Quareme

List<T>

- **Nadelen van arrays**
  - Lengte ligt vast, niet van grootte veranderen (tenzij Array class gebruiken)
  - Waarde toevoegen of verwijderen:
    - Kopie maken van array naar nieuwe array met nieuwe grootte.
    - Elke waarde opschuiven.

- **Oplossing**
  - Gebruik **List<T>** uit **System.Collections.Generic**
    - Kan je extra elementen aan toevoegen.
  - Gebruikt inwendig een array. Wanneer volle capaciteit ⇒ groeien
  - Is generiek
    - je kan het type T kiezen.
    - Als je zelf een generieke class of method maakt
      ⇒ geen aparte versie voor elk datatype nodig!

List<T>

- **Methods:**

Add(T element)	voegt element toe aan einde van List
AddRange(collection)	List/array toevoegen aan andere List
Clear()	wist alle elementen van List
Count	telt het aantal element in List (property)
ToArray()	kopieert List naar nieuwe 1D array
GetRange(int startIndex, int aantal)	maakt kopie van stukje van de lijst
IndexOf(T element) IndexOf(T element, int startIndex) IndexOf(T element, int startIndex, int aantal)	zoekt de index van een element uit List

List<T>

- **Methods:**

Insert(int index, T element)	voegt element toe op bepaalde index, de elementen erna schuiven 1 plaats op
Remove(T item)	verwijdert element met bepaalde waarde
RemoveAt(int index)	verwijdert element op bepaalde index
Reverse() Reverse(int startIndex, int aantal)	draait de volgorde van de elementen om
Sort()	sorteert de elementen in de List

- **Operator**
    - [ ] : om elementen op te zoeken of aan te passen (zoals in array)
      vb/ lijst[2] = 100; of int getal = lijst[3];

List<T> voorbeelden

- **Initialisatie (naïef, want List moet telkens inwendig gaan groeien)**

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

public class Program
{
    public static void Main()
    {
        int[] getallen = { 10, 20, 30, 40, 50, 60, 70, 80, 90, 100 };
        List<int> waarden = new List<int>(); // start met lege list, Count == 0
       // voeg waarden toe aan List (List verhoogt Count automatisch)
       foreach (int item in getallen)
            waarden.Add(item);

        Console.ReadLine();
    }
}
```

List<T> voorbeelden

- **Initialisatie (beter!!!) en elementen toevoegen aan List**

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

public class Program
{
    public static void Main()
    {
        // Hier hoeft de List niet meer te groeien.
        // List<int> waarden = new List<int>(new int[]{ 10, 20, 30, 40, 50, 60, 70, 80 });
        List<int> waarden = new List<int>{ 10, 20, 30, 40, 50, 60, 70, 80 };
        // Eventueel extra waarden toevoegen aan List achteraf
        waarden.Add(90);
        waarden.Add(100);

        Console.ReadLine();
    }
}
```

List<T> voorbeelden

- **AddRange() om array (of zelfs andere List) toe te voegen aan de List**

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

public class Program
{
    public static void Main()
    {
        int[] getallen = new int[]{ 10, 20, 30, 40, 50, 60, 70, 80 };
        List<int> waarden = new List<int>();

        // Voeg een array toe aan de List
        waarden.AddRange(getallen);

        Console.ReadLine();
    }
}
```

List<T> voorbeelden

- **Elementen inserten in List**

```
// Voeg waarde 1000 toe op index 3 van de List (indexes starten vanaf 0!!!)
// De waarden erna schuiven dan 1 plaats op
waarden.Insert(3, 1000);
```

- **Element verwijderen uit List**

```
// Verwijder element op index 3 van de List
waarden.RemoveAt(3);

// OF verwijder element met waarde 1000 (die als eerste voorkomt!) uit de List
waarden.Remove(1000);
```

List<T> voorbeelden

- **List omzetten naar array**

```
int[] waardenArray = waarden.ToArray();
```

- **Stukje van List omzetten naar array**

```
// Neem enkel het stukje van de List vanaf index 2 en 3 elementen als aantal
int[] waardenArray = waarden.GetRange(2, 3).ToArray();
```

List<T> voorbeelden

- **List sorteren**

```
string[] voornamen = { "Wouter", "Paul", "Andreas", "Niels", "Kathleen", "Paul", "Silvia",
    "Patricia"};

List<string> waarden = new List<string>(voornamen);

// waarden sorteren van klein naar groot
waarden.Sort();

// waarden sorteren van groot naar klein
waarden.Sort();
waarden.Reverse();

// eerste 3 elementen (start index: 0, aantal: 3) sorteren
// null is default comparer (van klein naar groot)
waarden.Sort(0, 3, null);
```

List<T> voorbeelden

- **Zoeken naar index in List**

```
waarden.IndexOf("Lise"); // => -1: niet gevonden
waarden.IndexOf("Paul", 3); // vanaf index 3 zoeken => 5 is het eerste voorkomen
waarden.IndexOf("Paul", 2, 2); // vanaf index 2 zoeken, aantal elementen: 2 => -1
```

Dictionary<K, V>

- **Dictionary (opzoektabel/woordenboek)**
  - Slaat elementen op als sleutelpaar (key-value paar) ⇒ 2 elementen.
    - Sleutel (key) van type K.
    - Waarde (value) van type V.

  - Met de sleutel kan je waarden opzoeken of verwijderen.
    - Kan je vergelijken als een soort index die eender welke waarde kan hebben of eender welk type.
    - Waardes kunnen ook eender welk type hebben.

  - **Gebruik:**
    - Als je vaak gegevens moet **opzoeken** (sneller dan (binair) zoeken in List).
    - Maar: Als je door gegevens moet *loopen* ⇒ array of List<> gebruiken.

Dictionary<K, V> voorbeelden

- **Dictionary aanmaken en sleutelpaar elementen toevoegen.**

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

public class Program
{
    public static void Main()
    {
        // Kies hier de datatypes van de key en value in deze volgorde.
        Dictionary<string, int> punten = new Dictionary<string, int>();

        // Voeg elementen toe aan dictionary: key is naam (string), value is punten (int)
        punten.Add("Tom", 18);
        punten["Sven"] = 20; // toevoegen kan ook op deze manier.

        Console.ReadLine();
    }
}
```

Dictionary<K, V> voorbeelden

- **Element opzoeken in dictionary**

```
// Element opzoeken in dictionary
int puntTom = punten["Tom"];
Console.WriteLine($"Tom heeft score: {puntTom}");

// Element PROBEREN op te zoeken in dictionary ⇒ geeft false terug als het niet bestaat
int puntAnna;
bool isGevonden = punten.TryGetValue("Anna", out puntAnna);
if (isGevonden)
    Console.WriteLine($"Anna heeft score: {puntAnna}");
```

Dictionary<K, V> voorbeelden

- **Element verwijderen uit dictionary**

```
punten.Remove("Sven");
bool isVerwijderd = punten.Remove("Sven");
```

- **Controleren of element bestaat in dictionary**

```
bool bestaat = punten.ContainsKey("Sven");
```

- **Loopen over dictionary (probeer dit te vermijden ⇒ trager dan loopen over array)**

```
// Loopen over dictionary en keys en values opvragen van elk item
foreach (var item in punten)
    Console.WriteLine($"{item.Key}: {item.Value}/20");
```

Dictionary<K, V> voorbeelden

- **Dictionary omzetten naar List (probeer dit te vermijden!!!)**

```
List<KeyValuePair<string, int>> list = dictionary.ToList();
// Daarna afdrukken
foreach (KeyValuePair<string, int> pair in list)
    Console.WriteLine($"{pair.Key}: {pair.Value}");
```