

<b>Module :</b>	PRB - Programming Basics	
<b>Opdracht :</b>	Lists oef 1	<b>Type :</b> Individuele Opdracht

# Namen en leeftijden

## DE OPDRACHT

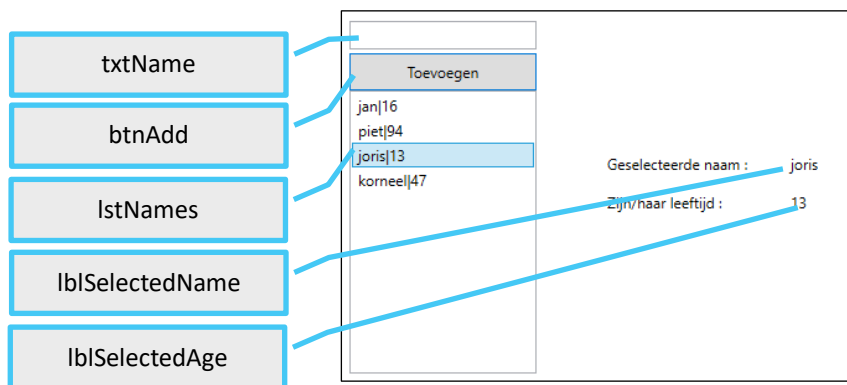


### Code repository

Haal de startsituatie van GitHub

`git clone` <https://github.com/howest-gp-prb/oe-lists-namenenleeftijden-start.git>

De controls :



De bedoeling :

- Telkens wanneer de gebruiker in txtName een naam invoert en op btnAdd klikt gaan we :
  - De naam bijhouden in een list (**names**) (= toevoegen aan de list)
  - Een leeftijd tussen 0 en 100 random (dus exclusief 0 en 100) laten genereren en bijhouden in een list (**ages**) (= toevoegen aan de list)
  - De naam, geconcateneerd met "|" en de leeftijd toevoegen aan de lstNames
- Indien in de naam het "|" zou worden gebruikt (wat weinig waarschijnlijk is) dient dit vervangen te worden door "I" (= hoofdletter i).
- We dienen er over te waken dat een naam niet 2 keer kan voorkomen.  
We kunnen dit realiseren door de Exists() methode van de list.

We kunnen dus iets schrijven als : `if (names.exists(<nieuwenaam> ) ...`  
(zie uitleg over predicaten hieronder).

- Wanneer we in de listbox op een item klikken dan :
  - Dient de waarde uitgelezen te worden
  - Dient de waarde “gesplit” te worden op het | -symbool.  
*Hiervoor kunnen we de string methode `split()` gebruiken.*

Voorbeeld :

```
char scheiding = '-';
string tekst = "tekst-met-koppeltekens";
string[] delen = tekst.Split(scheiding);
string eerste = delen[0];
string tweede = delen[1];
...
```

*1 van de (5) versies van de Split-methode gebruikt een karakter om een stuk te gaan opsplitsen in verschillende delen. De verschillende delen komen terecht in een array (hier dus de array “delen”).*

*Deze methode wordt pas echt bruikbaar voor ons van zodra we lussen behandeld hebben.*

*Nu is het voldoende voor deze oefening dat we weten dat de naam uiteindelijk in het eerste element van de array zal zitten.*

- Dient de positie van de naam opgezocht te worden in de list **names** (m.b.v. een Predicate, zie verder).
- Met het verkregen indexnummer (= positie) halen we de leeftijd van deze naam uit de list **ages**.

## Predicate

Heel wat methoden die je op een list kunt toepassen maken gebruik van een zgn Predicate.

Een goeie, eenvoudige website waar je alle methoden van de list (en het gebruik van predicaten) met voorbeelden kunt terugvinden : <http://www.csharp-examples.net/list/> .

Als je de betekenis van Predicate opzoekt, dan krijg je volgende antwoord :

Represents the method that defines a set of criteria and determines whether the specified object meets those criteria.

Eenvoudig uitgelegd : we moeten criteria opstellen (in ons geval : is de naam gelijk aan de zoeknaam?).

Afhankelijk van de gebruikte methode is de retourwaarde verschillend.

Bij de Exists-methode bijvoorbeeld wordt een boolean geretourneerd (gevonden of niet gevonden).

Bij de FindIndex-methode bijvoorbeeld wordt een integer geretourneerd (de eerste locatie waar de zoekwaarde gevonden werd).

Op zich allemaal logisch. Alleen de schrijfwijze is bij het eerste gebruik misschien een beetje vreemd.

```
if (names.Exists(zoek => zoek == naam))
```

```
{  
    ...  
}  
...  
int positie = names.FindIndex(zook => zoek == naam);
```

Het predikaat is hier dus : `zoek => zoek == naam`


Het criterium van het predikaat is de voorwaarde (de vraag) : `zoek == naam` (wel te verstaan, de variabele "naam" hebben we vooraf met een waarde gevuld).

Om predicate volledig te begrijpen dien je delegates te begrijpen, en dat wordt later in de opleiding uitgelegd. Het volstaat nu om te weten hoe je er gebruik van kunt maken.



#### Code repository

Klaar met deze oefening? Hier kan je de eindcode terugvinden

 `git clone` <https://github.com/howest-gp-prb/oe-lists-namenenleeftijden-einde.git>