Exercises about Linq

- Solve them in Visual Studio.

Exercise 22.01

Solve the following exercises with Linq fluent syntax.

- Create a list that contains integers.
- Show the list by looping thru it, and put a ";" in between. Make sure you don't have one at the end.
- Filter that list, so you have only the odd numbers.
- Show the list by looping thru it, using the same method as before.
- Sort that result in descending order.
- Show the list by looping thru it, using the same method as before.
- Calculate the average of all the numbers in the resulting list.
- Show result.

Variant 1 (Exercise 21.01)

- Rewrite this, using Query (Lambda) expressions.

Notes

.....

.....

.....

.....

.....

.....

.....

COPY PASTE)

Exercise 22.02

Solve the following exercises with Linq fluent syntax.

- Create a list that contains texts.
- Show the list by looping thru it, with a ";" in between.
- Filter that list, so you have only the texts that contain the letter "e", case insensitive.
- Show the list by looping thru it, using the same method as before.
- Sort that result in ascending order.
- Show the list by looping thru it, using the same method as before.
- Make the list in full capitals.
- Show the list by looping thru it, using the same method as before.

Variant 1 (Exercise 21.02)

- Rewrite this, using Query (Lambda) expressions.

Notes

.....

.....

.....

.....

.....

.....

.....

COPY PASTE)

Exercise 22.03

Solve the following exercises with Linq fluent syntax.

- Create a list that contains integers, there must be duplicates in it.
- Show the list by looping thru it, with a ";" in between.
- Filter that list, so you have only the even numbers.
- Show the list by looping thru it, using the same method as before.
- Create now a dictionary.
  - The key of the elements of the dictionary is the number itself.
  - The value of the elements it the number of times it occurs in the list of the even numbers.
- Sort this dictionary, on the frequency of the numbers. So you sort on the value of the dictionary. Highest number first. When there are equals, the dictionary must be sorted on the keys.
  - Think carefully on how you can do this.
- Show the result, every element of the dictionary on another line.
  - First the number (the key).
  - An arrow (-->).
  - Then the frequency (the value).

Variant 1 (Exercise 21.03)

- Rewrite this, using Query (Lambda) expressions.

Notes

.....

.....

.....

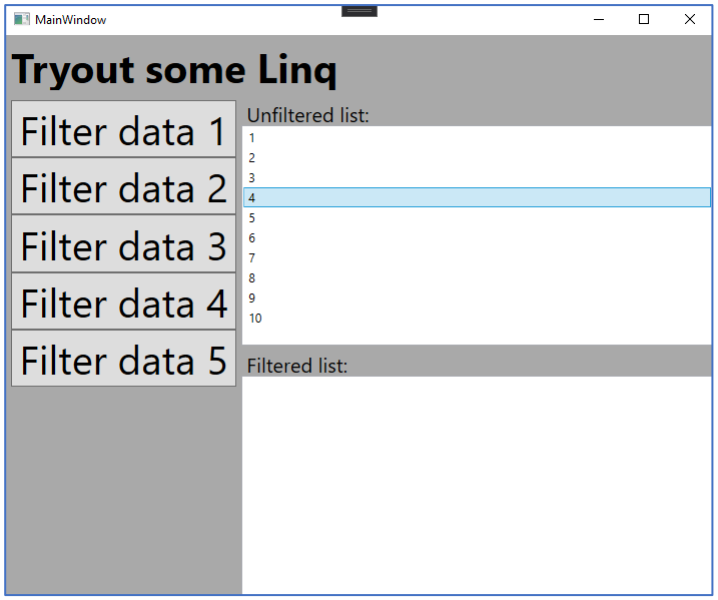
.....

.....

.....

.....

COPY PASTE)



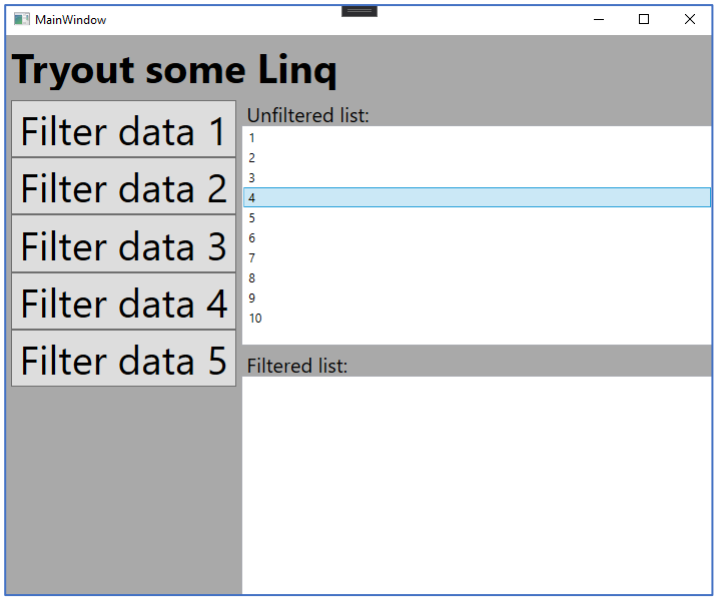
Exercise 22.04

- You create a WPF or a Windows Forms application that looks like beside.
- You have a unfiltered list, that contains some numbers.
  - You can experiment with strings, dates, whatever ….
  - The filter buttons should then filter on something else.
- You have a filtered list, that contains a part of the numbers of the unfiltered list.
- You have 5 buttons that filter the items from the unfiltered list, and put the result in the filtered list.
  - Make clear in the screen what it does.
- Button 1.
  - Show only the odd numbers.
- Button 2.
  - Show only the numbers that are bigger than 4.
- Button 3.
  - Show only the primes.
  - You have done this already in other exercises.
- Button 4.
  - Show only the numbers that can be divided by 3.
- Button 5.
  - Show only the numbers that can be divided by 3 and that are even.

Variant 1

- Rewrite this (Change Query (Lambda) Expression into fluent syntax, or vice versa)

Notes

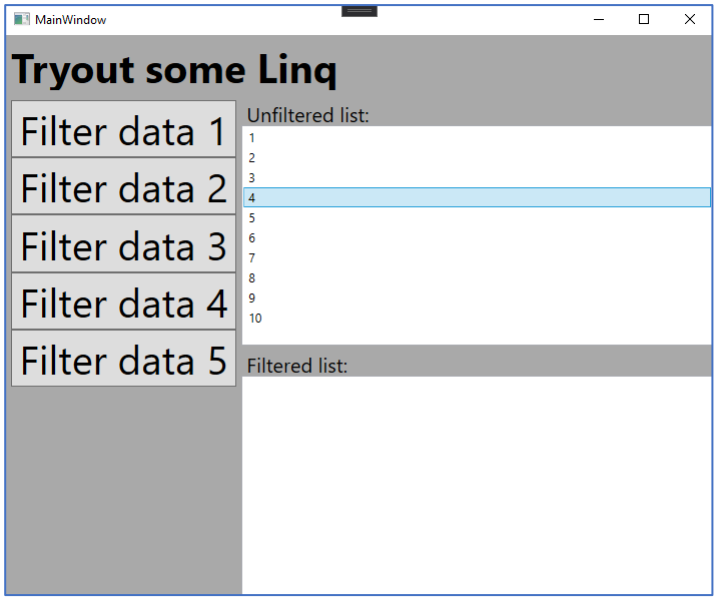


Exercise 22.05

- You create a WPF or a Windows Forms application that looks like beside.
  - Yes, it is the same as exercise 22.04 😊.
- You have a unfiltered list, that contains some students.
  - This is a whatever ….
- A student is a class that contains a Last name, a First name, an Age and a Gender (Male or Female).
- Use the class Student to create items in the unfiltered list.
- You have a filtered list, that contains another view of the students of the unfiltered list.
- You have 5 buttons that selects (projects) the items from the unfiltered list, and put the result in the filtered list.
  - Make clear in the screen what it does.
- Button 1.
  - Show the full name of the students.
- Button 2.
  - Show the students that are older than 40.
- Button 3.
  - Show the female students.
- Button 4.
  - Show all information of the students that contains a “e” in the first name and that are younger than 30.
- Button 5.
  - Show the students that have a longer first name than last name.

	*You can experiment with this by adding buttons to the screen.*

Notes



Exercise 22.06

- You create a WPF or a Windows Forms application that looks like beside.
  - Yes, it is the same as exercise 22.04 😊.
- You have a unfiltered list, that contains some students.
- A student is a class that contains a Last name, a First name, an Age and a Gender (Male or Female).
- Use the class Student to create items in the unfiltered list.
- You have a filtered list, that contains another view of the students of the unfiltered list.
- You have 5 buttons that selects (projects) the items from the unfiltered list, and put the result in the filtered list.
  - Make clear in the screen what it does.
- Button 1.
  - Show the number of students per gender.
- Button 2.
  - Show the names of students grouped by age (per decade)
  - 0 → 9, 10 → 19, 20 → 29, and so on ….
- Button 3.
  - Show the first name of students grouped by last name.
- Button 4.
  - Show the last names of students grouped by first name.
- Button 5.
  - Show the names of students grouped by gender.

You can experiment with this by adding buttons to the screen.

Variant 1: Rewrite this (Change Query (Lambda) Expression into fluent syntax, or vice versa).

Notes

COPY PASTE)

Exercise 22.07

Create a Console, WinForms or WPF application.

- Create a class “Training”.
- The class has several properties.
  - Teacher (Text).
    - Who is teaching the training.
  - Days (Number).
    - The number of training days is that are in the training.
  - Number of students (Number).
    - How many persons are following the training.
  - Period (Enum).
    - Before noon session, Afternoon session, Evening session.
    - All combinations are possible.
      - A training can be in the afternoon.
      - A training can be before noon and afternoon.
      - A training can be in the afternoon and evening.
      - And so on ….
- Create a number of instances of Training.
- Put them all in a list.
  - So you have a list of trainings.
- Create other lists that contain:
  - All trainings that are in the evening.
  - Count the number of training days for all the teachers.
  - Count the total of students per teacher.
  - Count the average of students for all trainings.
  - Count the average of students for all trainings per teacher.

Notes

Variant 1

- An extra (This is a difficult one).
  - Count all the sessions per training.
    - This is the number of days multiplied by the number of sessions that the training has.
    - Pay attention. The same training can be in the afternoon and the evening. So this are 2 sessions on the same day.
    - The day multiplied by the number of periods that are selected in the enum gives you the total number of sessions that are in the training.
  - You can choose how to solve this.
    - Either you have 3 groups and count trainings twice or triple.
    - Either you have different groups for all the combinations.
- An extra (This is also difficult, but is the same kind of problem)
  - If a student pays € 15 for every session for a training. How many € earns every teacher.
    - Take into account that a training day can be 1, 2 or 3 sessions.

Variant 2

- Rewrite this (Change Query (Lambda) Expression into fluent syntax, or vice versa).

Notes

Exercise 22.08

Create a Console, WinForms or WPF application.

- Create a class “NumberOfBedrooms”.
  - This contains an enumeration:
    - 1 bedroom.
    - 2 bedrooms.
    - 3 bedrooms.
    - 4 bedrooms.
    - 5 bedrooms.
- Create a class “Region”.
  - This contains an enumeration:
    - Antwerpen.
    - Limburg.
    - Oost-Vlaanderen.
    - Vlaams-Brabant.
    - West-Vlaanderen.
- Create a class “BuildingType”
  - This contains an enumeration:
    - Apartment.
    - Cottage.
    - Farm.
    - Villa.
    - House.
    - Separated house.
- Create a class “House”
  - This class has properties.
    - Name (Text).

Notes

.....

.....

.....

.....

.....

.....

.....

    - Region (Enumeration).
    - Price (Decimal).
    - Bedrooms (Enumeration).
    - Type (Enumeration).
- Create a number of instances of House.
- Put them all in a list.
    - So you have a list of houses.
- Create other lists that contain:
    - Number of buildings per region.
    - Number of average bedrooms per region.
    - Number of average price per region per type.
    - Number of average price per type per region.
    - Count the number of building per price (in steps of 25.000).
    - What is the total price of all houses.
    - What is the total price of the houses per region, sorted by highest region first.

Variant 1

- Rewrite this (Change Query (Lambda) Expression into fluent syntax, or vice versa).

Notes

Exercise 22.09

Create a Console, WinForms or WPF application.

- Create a class “Book”.
  - Properties:
    - An unique key.
    - Title.
    - Number of pages.
    - Writer.
    - Publisher.
- Create a class “Writer”.
  - Properties:
    - An unique key.
    - First name.
    - Last name.
    - Gender.
- Create a class “Publisher”.
  - Properties:
    - An unique key.
    - Name.
- Create demo data to create the Linq statements (in both syntaxes) below:
  - List all the books with the full name of the writer.
  - List all the books with the name of the publisher.
  - List all the books with the name of the publisher and the full name of the writer.
  - Count all the books per publisher and writer.
  - Count all the books per writer and publisher.

Notes

Exercise 22.10

Create a Console, WinForms or WPF application.

- Create a class “Product”.
  - Properties:
    - An unique key.
    - Weight of the product.
    - Price of the product.
    - Box. (the product is in this box)
- Create a class “Box”.
  - Properties:
    - An unique key.
    - Weight of the box.
    - Pallet (the box is on that pallet).
- Create a class “Pallet”.
  - Properties:
    - An unique key.
    - Client (is the client where the pallet is for).
    - Weight of the pallet.
    - Date of delivery.
- Create demo data to create the Linq statements (in both syntaxes) below:
  - Invent some Linq statements.
    - What is the full weight of the pallet?
    - Give me a number of products going to every client.
  - You can do whatever you want, but the application must make clear what you are doing.

Notes