Dag 5 Øvelser

Øvelse 5.1

I et nyt projekt, i Main-metoden, lav en liste af ints.

Lav en søgning på alle lige ints (tal % 2 skal give 0), udskriv resultatet vha. ForEach metoden på listen. Dvs. du skal lave en Action ligesom i eksemplet:

```
resultat.ForEach(k => Console.WriteLine(k.Klassenavn));
```

Lav en søgning efter det sidste tal over 15 (så du skal nok have mindst et over 15)

Lav en søgning efter index på det sidste tal over 15 (så du skal nok have mindst to over 15)

Øvelse 5.2

I samme projekt som før, lav nedenstående søgninger vha LinQ. Prøv både med Query Methods og Query Expressions.

Lav følgende søgninger:

- Alle lige ints
- Alle ints med præcis to cifre (brug tostring og længden)

Bemærk, her kan man ikke bruge ForEach-metoden, da resultatet ikke er en List, men en IEnumerable.

Sorter resultatet og udskriv igen

I de næste øvelser vil du træne at håndtere data – både uden LINQ men også med LINQ. Normalt ville man have en database, men for nu er data gemt i en .csv fil som vi kan indlæse. Du kan se data ved at åbne .csv filen i f.eks. Excel, hvis du vil have et overblik. Hver række repræsenterer en person med et navn, en alder og en vægt og en score (et antal point) – i den rækkefølge som beskrevet her.

Lav et nyt consol project til disse øvelser.

Øvelse 5.3 Using FindAll

Lav en Person klasse med følgende Properties:

```
public string Name { get; set; }
```

```
        public int Age { get; set; }
        public int Weight { get; set; }
        public int Score { get; set; }

        public bool Accepted { get; set; }
```

og lav også en ToString som skriver disse ting ud for en Person.

Load alle personerne fra data1.csv ind i en List: List<Person>

Du kan gøre noget i stil med dette (bemærk – du skal nok ændre din path afhængig af hvor du lægger din .csv fil):

```
public static void Exercise1() {
            try {

                people1 = Person.ReadCSVFile(@"C:\Users\makn\Desktop\dmu_csharp\MK - 2021
efterår\Dag 05 LINQ\Løsninger\SearchAndSortInData\SearchAndSortInData\data1.csv");

            }
            catch (Exception ex) {
                Console.WriteLine("EXCEPTION: " + ex.Message);
                Console.WriteLine("You should probably set the filename to the
Person.ReadCSVFile method to something on your disk!");

            }
```

(hvor READCSVFILE er defineret således i en Person klasse):

```
public static List<Person> ReadCSVFile(string filename) {
            List<Person> list = new List<Person>();
            using (var file = new StreamReader(filename)) {
                string line;
                while ((line = file.ReadLine()) != null) {
                    var p = new Person(line);
                    list.Add(p);
                    //Console.WriteLine(p);
                }
            }
            return list;

        }
```

Du skal så have en constructor som splitter en linje i de forskellige elementer:

```
public Person(string data) {
            // Name, Age, Weight, Score
            var L = data.Split(';');

            Name = L[0];
            Age = int.Parse(L[1]);
            Weight = int.Parse(L[2]);
            Score = int.Parse(L[3]);
            Accepted = false; //denne parameter bliver ikke indlæst, men skal bruges senere.

        }
```

Hvor people1 er defineret således:

```
List<Person> people1
```

Når data er loadet ind så brug FindAll method på List<T> til at lave 4 forskellige filtreringer:

1. Alle personer med en score under 2
2. Alle personer med en lige score, altså 0, 2, 4, 6…
3. Alle personer med en lige score og weight større end 60
4. Alle personer med en vægt deleligt med 3

Hint: Du kan bruge modulus operatoren, %, til at finde ud af om noget kan deles med et nummer. F.eks. er p.score % 4 == 0 udtrykket sand, hvis scoren kan deles med 4 (uden rest altså, derfor 0).

Du kan se dokumentationen for FindAll her:

https://docs.microsoft.com/en-us/dotnet/api/system.collections.generic.list-1.findall?view=net-5.0

Bemærk at FindAll tager en `Predicate<T> match` som parameter, så dette skal du opfylde.

Det vil være nemmest at bruge lambda udtryk til disse øvelser!!

Skriv resultatet ud og se om det passer med data for at teste om du har skrevet koden korrekt.

Øvelse 5.4 using FindIndex

Dokumentationen for FindIndex kan fines her : https://docs.microsoft.com/en-us/dotnet/api/system.collections.generic.list-1.findindex?view=net-5.0

(alle queries med FindIndex skal du skrive som lambda udtryk - nemmest)

1. Brug FindIndex metoden på listen til at finde index på den første person med en score på præcis 3.
2. Brug FindIndex til at finde index på den første person under 10 år, som har en score på 3.
3. Hvor mange personer er der under 10 år som har en score på 3? (Hint: Her skal du bruge FindAll…)
4. Brug FindIndex til at finde index på den første personer under 8 år, med en score på 3. Bemærk return value her – hvad betyder dette?

Øvelse 5.5 Sorting with List<T>.Sort()

Nedenstående opgave kan du springe over, du har lavet tilsvarende tidligere, den er mest med for at illustrere forskellen til Linq lidt senere.

Sorter personer efter både Score og Age, (både ascending og descending).

Du kan lave en ny klasse som implementer ICompare interfacet og så bruge den klasse som argument til sort: F.eks. `public class SortByAge : IComparer<Person> {…}`

```
Og så kan du sortere sådan her:   people.Sort(new Person.SortByAge());
//her er SortByAge klassen defineret inde i Person klassen
```

Øvelse 5.6 – predicates og extension methods

Skriv en extension metode på List<Person> Klassen så du kan skrive følgende kode :

people.SetAccepted(p => p.Score >= 6 && p.Age <= 40);

(Hint: din extension metode skal have signaturen

```
        public static void SetAcceptedP(this List<Person> lst, Predicate<Person> pred)
```

)

Men hvad skal den så gøre i kroppen? Ja, den skal jo finde alle personer som opfylder predikatet, og så sætte accepted til true på alle de personer….Så der er nok noget med FindAll og Foreach på resultatet fra FindAll….

Efter dit kald til people.setAccepted, så skriv listen af personer ud og se om acceptet er blevet sat på de personer som opfylder predikatet – det er jo testen på om dit kode virker!

Øvelse 5.7 LINQ

Brug LINQ til at sortere listen af personer efter Score og Age. Sortér efter både stigende og faldende orden. Bemærk, at nu skal vi ikke til at implementere et interface eller noget…så koden burde være ”pænere”. Prøv at lave det med Query Expression syntaxen – det er nok det nemmeste at overskue. Se slides for eksempler på syntaxen.

Øvelse 5.8 LINQ

Lav et array af tal :

```
int[] numbers = { 34, 8, 56, 31, 79, 150, 88, 7, 200, 47, 88, 20 };
```

Skriv forskellige LINQ statements til at lave de følgende ting:

1. Returner alle two-digit integers sorteret i ascending order.
2. Returner alle two-digit integers sorteret i descending order.

3. Som i delopgave 1), men I stedet for integers, så skal der returneres strings dvs. f.eks. ”20”, ”31”, ”34”, etc.
4. Som i delopgave 2), men skal returnere string af typen “20 even”, “31 uneven”, etc…

Hint: til delopgave 4 her, får du nok brug for en conditional operator, da return værdier jo skal afhænge af om det er et lige eller et ulige tal.

https://docs.microsoft.com/en-us/dotnet/csharpæ/language-reference/operators/conditional-operator (Links til en ekstern webside.)

Øvelse 5.9

Skriv en extension method på List<Person>, sådan at du kan lave følgende metodekalde direkte på listen af personer:

```
people.Reset();
```

Metoden skal sætte person.Accepted = false for alle Personer i listen.

Øvelse 5.10

Lav en liste af 100 random integers. Så brug LINQ til de følgende delopgaver:

1. a) Find antallet af ulige tal I listen.
2. b) Find antallet af unikke tal I listen (f.eks. skal 3,3,5 give 2)
3. c) Find de tre første ulige tal.
4. d) Find alle unikke ulige tal.

Hint: Det er nok nemmest først at udvælge elementer, og så bagefter tælle hvor mange unikke numre der er. Dette kan f.eks. gøres på en liste sådan her:

```
int distinct = numbers.Distinct().Count(); //hvor numbers er en liste fra en LINQ query
```

Øvelse 5.11 – ekstra opgave

Brug LINQ til at gruppere alle personerne efter det første bogstav I deres navn, altså Anders og Anita skal være i samme gruppe og Bent og Bo skal være i samme gruppe.

Du kan få brug for dette her:

https://docs.microsoft.com/en-us/dotnet/csharp/linq/group-query-results (Links til en ekstern webside.)

Øvelse 5.12 – extra opgave

Skriv LINQ statements som finder alle personer, dvs. med samme navne, som er både i data1.csv og data2. Dette er sådan set hvad vi ville kalde en inner join på to tabeller i databasesprog. Se evt. https://www.tutorialsteacher.com/linq/linq-joining-operator-join

(både data1.csv og data2.csv kan findes på Canvas)