Sovelto

Viikko 2, LINQ, kokoelmat

Kontrollirakenteita (kertausta ja opitun vahvistamista)
Luokat ja kokoelmat
LINQ

ACADEMY
A PART OF ACADEMIC WORK

Jos tänään joku oppilas oppii jonkin asian niin päivä on onnistunut

Laskentaa

- Tee konsolisovellus joka tulostaa kertotaulun
- Tee uusi projekti (Console Application)
- Main-metodissa kysy käyttäjältä luku, lue rivi Console.WriteLine:llä ja muuta se kokonaisluvuksi
- Tulosta luvun kertotaulu, ohessa esimerkki jos käyttäjä on antanut luvun 4

```
 1 * 4 = 4
 2 * 4 = 8
 3 * 4 = 12
 4 * 4 = 16
...
 9 * 4 = 36
10 * 4 = 40
```

- Lisätehtävä: Lisää try/catch eli poikkeuskäsittelijä siten että sovellus antaa asiallisen ilmoituksen jos numeroa ei voi muuttaa kokonaisluvuksi
- Lisätehtävä 2: tulosta lopuksi annetun luvun kertoma, esim neljän kertoma (merkitään 4!) on 1 * 2 * 3 * 4

Merkkijonoja ja taulukoita

- Oheisessa taulukossa on kurssin konsulttien nimet
- Tee sovellus joka tulostaa taulukossa olevat nimet ensin etunimen mukaan aakkosjärjestyksessä (konsolisovellus on hyvä tähän harjoitukseen). Ota copy/paste-menetelmällä oheinen taulukko sovellukseesi.
- Entä sitten tulostus sukunimen mukaan järjestyksessä? Tämä ei ole ihan helppo parilla rivillä tehtävä ratkaisu, jätetään odottamaan LINQ-asioita

```
string[] konsultit =
{
  "Muhsen Almasry",
  "Juri Heiniluoma",
  "Heidi Vähätalo",
  "Niina Siitari",
  "Riku Soikkeli",
  "Ari Savolainen",
  "Jani Huusko",
  "Matias Vähäkangas",
  "Emma Saarelainen",
  "Marko Salmikangas",
  "Johanna Niklander ",
  "Janica Jokela",
  "Noora Lohi",
  "Thien Nguyen",
  "Otso Peippo",
  "Roope Rouvali"
};
```

Konsultit, edelliseen jatkotehtävä (kokoelma)

- Lisää edelliseen sovellukseen luokka class Konsultti jossa on Etunimi ja Sukunimi – ominaisuudet, voit tehdä myös konstruktorin mutta ilman sitä pärjätään
- lisää sovellukseesi kokoelma
  List<Konsultti> konsulttilista ja lisää siihen kaikki taulukossa olevat konsultit
  - käy taulukko läpi ja Split-metodilla pura etunimi ja sukunimi erilleen, sitten luo Konsultti-olio, aseta ominaisuudet ja lisää olio konsulttilista-kokoelmaan
- Nyt sinulla on huomattavasti joustavampi tietorakenne konsulttien käsittelyyn ja tähän osaan palataan LINQ-asian yhteydessä johon tässä on oivallinen kokoelma
- Seuraavaksi lisätään sovellukseen toiminto joka arpoo konsultit ryhmiin, olkoot aluksi ryhmäkoko esimerkiksi 5 konsulttia
- Käytä Random-luokkaa ja tulosta konsultit arvottuihin ryhmiin
- Lisää vinkkejä seuraavalla sivulla
- Lisätehtävä: tee lisäominaisuus joka arpoo päivittäisen istumajärjestyksen

Vinkkejä

```
// jossain on kokoelma List<Konsultti> konsulttilista; ja se sisältää kaikki
// konsultit, toivottavasti tämä on jo tehtynä
Random rnd = new Random(); // satunnaislukugeneraattori
int ryhmäKoko = 5; // tämä siis toistaiseksi kovakoodattuna
int ryhmienLkm = konsulttilista.Count / ryhmäKoko; // montako ryhmää
for (int i = 0; i < ryhmienLkm; i++) { // näin monta ryhmää pitäisi arpoa
  // esimerkiksi näin:
  // tulosta tässä esim.: Ryhmä 1:
  for (int j = 0; j < ryhmäKoko; j++) {
        // arvo satunnaisluku välillä 0 – (konsulttilista.Count – 1)
        // tulosta kyseisen konsultin nimi konsolille
        // poista tämä konsultti kokoelmasta, RemoveAt tai Remove -metodilla
  }
  Console.WriteLine();
  // selvitä itsellesi miten edellä esitetty algoritmi toimii
  // keksitkö jonkin muun tavan tehdä arvonta?
}
```

Vinkkejä – toteutus SortedDictionary-luokalla

```
Random rnd = new Random(); // satunnaislukugeneraattori
// luo SortedDictionary jossa arvot ovat avaimen mukaisessa järjestyksessä
SortedDictionary<int, Konsultti> sd = new SortedDictionary<int, Konsultti>();
int ryhmäKoko = 5; // tämä siis toistaiseksi kovakoodattuna
int ryhmienLkm = konsultit.Length / ryhmäKoko; // montako ryhmää
// käy läpi konsultit-taulukko ja silmukassa teet seuraavaa
foreach (int k in konsultit) {
  // arvo luku välillä 1-1000
  // lisää sd-kokoelmaan konsultti niin että avain (key) on juuri arpomasi luku
  sd.add(arvottuluku, k); // huonolla tuurilla kaatuu mutta miksi?
}
// nyt konsultit ovat 'arvottu' johonkin järjestykseen
// käy läpi sd-kokoelman kaikki avaimet ja tulosta avaimen mukaisessa
// järjestyksessä ryhmät
// foreach(var avain in sd.Keys) {
//       Konsultti kon = sd[avain]; // jotenkin näin
```

Extension Methods

- String-luokkaa ei voi muuttaa ja siitä puuttuu metodi jolla voisi poistaa kaikki ylimääräiset välilyönnit merkkijonon sisältä, esimerkiksi jos käyttäjä syöttää nimensä muodossa
  Aku Ankka tai osoitteen Keilaranta 42 niin näistä olisi hyvä siistiä turhat blankot pois ennen tietokantaan talletusta

1. Tee extension-metodi BlankkoTrimmaus joka poistaa turhat välilyönnit, tässä voit käyttää esimerkkinä Palindromi-harjoituksessa ollutta esimerkkiä
2. tee uusi static luokka, siihen static metodi ja ensimmäinen parametri this –määrellä
3. testaa toiminta, esim:
   string s = "tes ti mjo no";
   s = s.BlankkoTrimmaus();

```
...
public static class ExtensioHarjoitus {
  public static string BlankkoTrimmaus(this string s) {
            // ja tässä sitten poistat ylimääräiset välilyönnit
}
```

4. Lisätehtävä:
   Lisää uusi parametri
   params char[] merkit
   ja tämän avulla välität siis ne merkit jotka pitäisi poistaa merkkijonon sisältä

LINQ

- Käytä harjoitukseen List-kokoelmaa jossa on Konsultti-oliot
- tässä harjoituksessa käytetään pelkästään LINQ:n ominaisuuksia ja tulostus konsolille

1. tulosta ne konsultit joiden nimessä on "nen" ja sitten ne joiden nimestä puuttuu "nen"
2. tulosta konsultit sukunimen mukaan aakkosjärjestyksessä
3. tulosta konsultit etunimen mukaan takaperoisessa aakkosjärjestyksessä
4. tulosta konsultin tiedot jonka etunimen ja sukunimen yhteenlaskettu pituus on suurin
5. tulosta konsultin tiedot jonka sukunimessä on vähiten vokaaleja
6. hae kaikki konsultit ja muodosta niistä uusi anonyymiluokka jossa on nimi ja nimen pituus, nimi muodossa Sukunimi, Etunimi ja tulosta konsolille
7. kokeile onnistutko kirjoittamaan kyselyt Extension Methods –muotoon?