HALADÓ PROGRAMOZÁS

LINQ

LINQ (Language Integrated Queries)

- Gyűjtemények szintaktikailag egyszerű és forrásfüggetlen kezelését biztosítja
  - Szintaktikailag egyszerű: (ciklusokkal, elágazásokkal megoldható) gyakori feladatok megoldását biztosító „operátorok”
  - Forrásfüggetlen: tömb, lista, XML, adatbázis ugyanolyan módon kezelhető
- LINQ To Objects, LINQ To XML, LINQ to Entities, ...
- Részei és támogató technológiák:
  - LINQ operátorok
  - LINQ lekérdezések kulcsszavai
  + Lambda kifejezések
  + var kulcsszó és névtelen osztályok
  + Kiegészítő/bővítő metódusok

LINQ

- Gyűjteményeken sokszor műveleteket végez,
- például rendezi a gyűjtemény elemeit,
- szűréseket végez a gyűjteményen
- kettő (vagy több) gyűjteményt összekapcsol egy lekérdezésen belül (join)
- *LINQ (Language Integrated Query)*
  - *LINQ to Objects*
  - *LINQ to XML*
  - *LINQ to Entities*
  - *LINQ to SQL*

LINQ előnyei

- LINQ **szintaxis színezést** lehetővé teszi, könnyű a hibákat tervezési időben kiszűrni.
- LINQ használja az IntelliSense **súgó**t.
- Sokkal gyorsabb a kódolás .
- LINQ lehetővé teszi az egyszerű **debug**golást, mivel integrálva van a C# nyelvbe.
- Könnyű a **különböző táblákat** összekapcsolni.
- LINQ ugyanazt a szintaktikát használja egészen különböző adatforrások lekérdezéséhez.
- LINQ tovább bővíthető újabb és újabb adatszerkezetekhez
- LINQ egy lekérdezésen belül is többféle adatforrás használatát teszi lehetővé.
- LINQ megkönnyíti a konverziót pl. SQL és XML között.

HÁTRÁNYA

Lassabb végrehajtás

LINQ vs Hagyományos megközelítés

```
static void QueryOverStringsLongHand()
{
  // Assume we have an array of strings.
  string[] currentVideoGames = {"Morrowind", "Uncharted 2", "Fallout 3", "Daxter", "System
  Shock 2"};

  string[] gamesWithSpaces = new string[5];

  for (int i = 0; i < currentVideoGames.Length; i++)
  {
    if (currentVideoGames[i].Contains(" "))
    {
      gamesWithSpaces[i] = currentVideoGames[i];
    }
  }

  // Now sort them.
  Array.Sort(gamesWithSpaces);

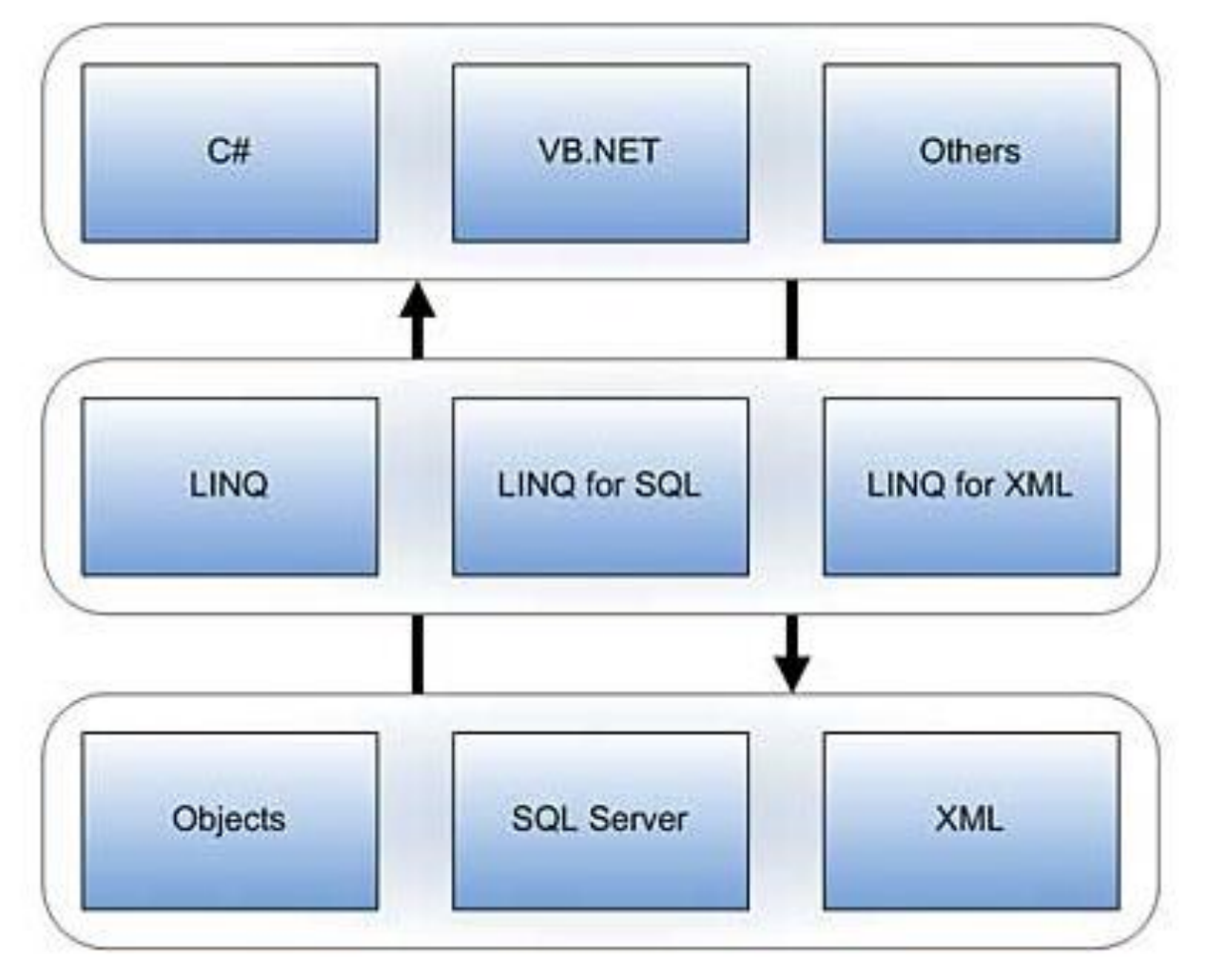
  // Print out the results.
  foreach (string s in gamesWithSpaces)
  {
    if( s != null)
    {
      Console.WriteLine("Item: {0}", s);
    }
  }
```

```
static void QueryOverStringsWithExtensionMethods()
{
  // Assume we have an array of strings.
  string[] currentVideoGames = {"Morrowind", "Uncharted 2", "Fallout 3", "Daxter", "System
  Shock 2"};

  // Build a query expression to find the items in the array
  // that have an embedded space.
  IEnumerable<string> subset =
    currentVideoGames.Where(g => g.Contains(" ")).OrderBy(g => g).Select(g => g);

  // Print out the results.
  foreach (string s in subset)
  {
    Console.WriteLine("Item: {0}", s);
  }
}
```

LINQ szerkezete



LINQ operátorok

- IEnumerable<T> interfész új metódusai (meglévő régiek mellé definiálták őket)
  - … Használhatóak tömbön, listán, egyebeken (IEnumerable<T>-t implementáló típusokon)
  - +IQueryable<T>
- Nagy csoportok azonosíthatók:
  - **Elemkiválasztás** – első, utolsó, egyetlen, …
  - **Szűrés** – tulajdonságnak megfelelő…
  - **Rendezés** – előrefelé, megfordítás, …
  - **„Halmaz”kezelés** – unió, metszet, …
  - **Aggregáció (~számítások)** – max, min, …
  - **Csoportosítások**
- IEnumerable<T>-n dolgoznak, a kimenet többnyire:
  - IEnumerable<T> vagy IEnumerable<X>, emiatt **láncolhatók**
  - T vagy X
- Nagyon sokszor metódussal paraméterezhetők

Nyelvi elemek – var és névtelen osztályok

- var: deklarált változó típusának meghatározását a fordítóra bízzuk
  - Kötelező azonnali értékadás

```
var x = 6;
var z = new Hallgato();
```

- névtelen osztályok: ideiglenes, apró, csak adatot összefogó osztályok helyett

```
var nevKorCim = new
{
    Nev = "Béla",
    Kor = 23,
    Cim = "Budapest Bécsi út 96/B"
};
```

LINQ operátor példák

```
int[] elso = new int[] { 2, 4, 6, 8, 2, 1, 2, 3 };
int[] masodik = new int[] { 1, 3, 5, 7, 1, 1, 2, 3 };
string[] strtomb = new string[] { "Béla", "Jolán", "Bill", "Shakespeare",
  "Verne", "Jókai" };
List<Diak> diakok = new List<Diak>();
diakok.Add(new Diak("Első Egon", 52));
diakok.Add(new Diak("Második Miksa", 97));
diakok.Add(new Diak("Harmadik Huba", 10));
diakok.Add(new Diak("Negyedik Néró", 89));
diakok.Add(new Diak("Ötödik Ödön", 69));
```

LINQ operátor példák – halmazok

Elem létezésének vizsgálata:

```
bool bennevan=elso.Contains(4);
```

Két gyűjtemény egymás után fűzése (NEM halmazok!):

```
var uj = elso.Concat(masodik);
```

Ismétlődések kivágása (halmazzá alakítás):

```
var uj = elso.Distinct();
```

Halmazelméleti metszet:

```
var uj = elso.Intersect(masodik);
```

Halmazelméleti unió:

```
var uj = elso.Union(masodik);
```

Halmazelméleti különbség

```
var uj = elso.Except(masodik);
```

LINQ operátor példák – sorrendezés

- **OrderBy**
  - **Paraméterül egy olyan eljárást vár, amely egy osztályból kiszedi a kulcsot (azt a mezőt, ami alapján rendezni fog) (Ehelyett egy lambda kifejezést szokás írni)**
  - ***Második paraméterként megadható neki egy saját, IComparer interfészt implementáló osztály, ami az összehasonlítást végzi***
  - **Int tömb, rendezés az elemek alapján:**

```
var uj = elso.OrderBy(x => x);
```

  - **String tömb, rendezés az elemek hossza alapján:**

```
var uj = strtomb.OrderBy(x => x.Length);
```

  - **Diákok listája, névsorba rendezés :**

```
var uj = diakok.OrderBy(x => x.Nev);
```

LINQ operátor példák – szűrés, darabszám

- Where / Count
  - **A paraméterül adott kifejezésnek bool típust kell visszaadni.**
  - **A *Where* eredménye az a gyűjtemény, ahol ez *true* értéket ad vissza. A *Count* eredménye a darabszám (int!!)**
  - **A count meghívható paraméter nélkül is ➔ teljes darabszám**
  - **Int tömb, a páratlanok:**

```
var uj = elso.Where(x => x % 2 == 0);
```

  - **String tömb, a négy betűs nevek:**

```
int num = strtomb.Count(x => x.Length == 4);
```

LINQ operátor példák – szűrés, részkiválasztás

- Diákok listája, csak név:

```
var uj = diakok.Select(x => x.Nev);
```

- Diákok listája, ahol a kreditszám prím:

```
var uj = diakok.Where(x =>
{
   if (x.Kreditek==1) return false;
   for (int i = 2; i <=
 Math.Sqrt(x.Kreditek);
      i++) if (x.Kreditek % i == 0) return
 false;
   return true;
});
```

LINQ operátor példák – láncolás

- Diákok listája, a páratlan kreditszámúak nagybetűs neve név szerinti fordított sorrendben:

```
var uj = diakok.Where(x => x.Kreditek % 2 == 1)
  .OrderBy(x => x.Nev)
  .Reverse()
  .Select(x => x.Nev.ToUpper());


// ÖTÖDIK ÖDÖN, NEGYEDIK NÉRÓ, MÁSODIK MIKSA
```

- Ugyanaz az eredmény, ugyanaz a köztes kód:

```
var uj = from x in diakok
      where x.Kreditek % 2 == 1
      orderby x.Nev descending
      select x.Nev.ToUpper();
```

LINQ operátor példák – aggregálás

- Aggregáló metódusok

```
int ossz = elso.Sum(); //28
double atlag = masodik.Average(); //2.875
int parosOssz = elso
  .Where(x => x % 2 == 0).Sum(); //24
int paratlanOssz = elso
  .Where(x => x % 2 == 1).Sum(); //4
```

- A fenti példa gyakori: valamilyen ismétlődés szerint akarom csoportosítani a gyűjteményemet, és az egyes csoportokra szeretném tudni a darabszámot/összeget
  - Csoportonként egy where+aggregálás ➔ zavaró ➔ automata csoportosítás: GroupBy

Kétféle formában használhtó

- Bővítő metódusok
  - Where, Orderby stb.
  - Paraméterei …=>… lambda kifejezések
- Alternatív forma- lekérdező kifejezés

```
IEnumerable<string> citiesToDisplay
= from c in cities where c.Length
>= 8 orderby c.Length select c;
```

Bővítő metódusok

```
IEnumerable<DateTime> datesToDisplay = dates.Where(d =>
d.Year > 2000);
```

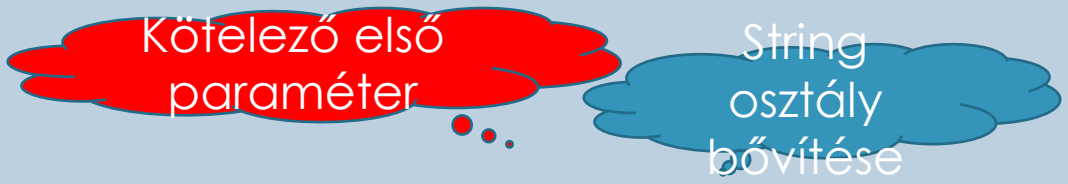
- dates objektumnak rengeteg a Where-hez hasonló, hasznos műveletet végző metódusa van, pl. keresésre, rendezésre, kiválasztásra, összegzésre stb.
- Ezek nem is a List osztály, hanem az IEnumerable interfész metódusai.
- IEnumerable a valóságban csupán egyetlen metódust definiál (GetEnumerator).

Bővítő metódus fogalma

- IEnumerable-n kívül, más osztályokban is lehet megírni,
- és kívülről bővíteni velük a IEnumerable-t.
- Public, static osztályban lehet bővítő metódust definiálni

```
public static class MyExtensionMethods
{

public static int CountChar(this string s, char c)
{
int count = 0;
foreach (char x in s)
if (x == c) count++;
return count;
} }
```



Fontos

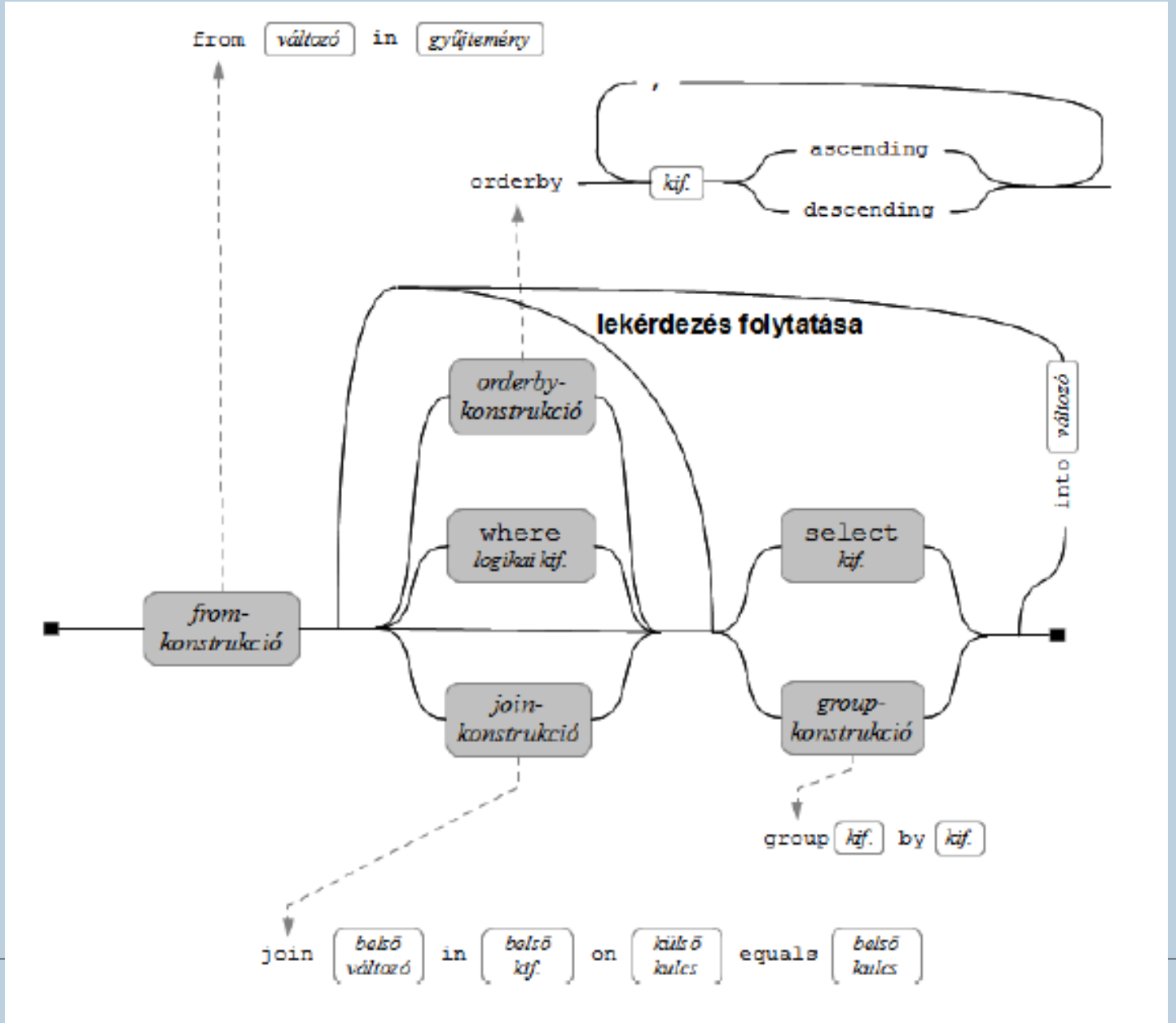
- Bővítő metódusok használatához szükséges a LINQ névtér
- using System:Linq
- System.Linq névtér Enumerable (statikus) osztályában vannak definiálva

Lekérdező szintaxis

```
public static IEnumerable<T> Where<T>(this IEnumerable<T> source,
Func<T, bool> predicate);
```

- Mint látható, az első paraméter előtt **this** kulcsszó áll, azaz ez a metódus az IEnumerable<T> interfészt bővíti. A második paraméter típusa egy (a System névtérben definiált) **delegált**, ami tetszőleges típusú (T) paramétert vár, és bool típussal tér vissza.
- LINQ késleltetett végrehajtású

Lekérdező szintaxis szerkezete



Késleltetett végrehajtás

```
List<int> numbers = new List<int>() { 1, 2 };
IEnumerable<int> query = numbers.Select(n => n * 10);
numbers.Add(3);
foreach (int n in query)
textBlock.Text += string.Format("{0} ", n);
```

Eredmény: 10,20,30

LINQ operátor példák – csoportosítás

- Csoportosítás, paritás szerinti darabszámok:

```
var csoport=elso.GroupBy(x => x % 2);
foreach (var g in csoport)
{
    Console.WriteLine("Maradék: " + g.Key +
            ", darabszám: " + g.Count());
}

var uj=from x in elso
   group x by x%2 into g
   select new {Maradek=g.Key, Darab=g.Count()};
```

Haladó Programozás

LINQ bevezetés, LINQ to Objects
XML kezelés, LINQ to XML
Feladat

XML emlékeztető (w3schools.com)

```
<?xml version="1.0"?>
<bookstore>
  <book category="COOKING">
    <title lang="en">Everyday Italian</title>
    <author>Giada De Laurentiis</author>
    <year>2005</year>
    <price>30.00</price>
  </book>
  <book category="WEB">
    <title lang="en">Learning XML</title>
    <author>Erik T. Ray</author>
    <year>2003</year>
    <price>39.95</price>
  </book>
</bookstore>
```

- Hierarchikus adatleíró formátum
- XML deklarációk + elemek + attribútumok
  - Elemek: <bookstore></bookstore>, <book></book> = <tag></tag>
  - Attribútumok: <book>-ban category=„…” …

XML emlékeztető

- Felépítését szigorú szabályok korlátozzák
  - Első sor: opcionális formátumspecifikáció, kiegészíthető karakterkódolással:
    <?xml version="1.0" encoding="ISO-8859-1"?>
    <?xml version="1.0" encoding="UTF-8"?>
  - **Mindegyik elemnek lehet:**
    - **Szöveges tartalma**
    - **Alelemei**
    - **Attribútumai, amelyek az adott elemhez adnak meg tulajdonságokat**
  - **Kötelezően kell lennie egy gyökérelemnek, ami az egész dokumentumot közrefogja (<bookstore> elem)**
  - Minden elem lezárása kötelező (<tag></tag> vagy <tag />)
    - Az egymásba ágyazás lezárásainak megfelelő sorrendben kell történniük
    - Rosszul formázott: <a>
    - Jól formázott: <a>

XML emlékeztető – attribútum vs. alelem

- Bár szabály nincs rá, de a józan ész elvei szerint kell használni az attribútumokat és a gyerekelemeket. Ez a megoldás szintaktikailag helyes lenne, de nem ésszerű:

```
<note day="10" month="01" year="2008"
to="Tove" from="Jani" heading="Reminder"
body="Don't forget me this weekend!">
</note>
```

- Javaslat: attribútum csak kivételes esetben, legtöbbször csak az id=„0001", ami alapján az elemek azonosíthatók

XML emlékeztető – szabályok

- A kis- és nagybetűk különbözőek
- Az attribútumok jelzésénél kötelező az idézőjel vagy az aposztróf
- Speciális karakterek helyett entity-ket használunk:

Karakter	Entity
<	<
>	>
&	&
‘	'
"	"

XML emlékeztető – szabályok

- <!-- megjegyzés -->
- A szöveges tartalmakban a szóköz karakterek megmaradnak
- Sorvégjel: linux/unix szerű \n (\r nem kell, de a legtöbbször nem gond)
- Tag-névben lehet szám és betű is, de többnyire csak az angol ABC betűit használjuk
  - **Lehetőleg legyen rövid**
  - **Space nem használható, inkább _**
  - **Pont, kötőjel, kettőspont nem javasolt, inkább _**
  - **Szám nem lehet a név elején**
  - **DTD/Schema készíthető: „well formed” vs „valid”**
- Az XML kiegészíthető: új tag bevezetése nem teszi tönkre az eddig használt feldolgozó programokat

XML + .NET

- Háromféle XML-technológia
- XmlReader, XmlWriter
  - **Gyorsak, kevés memóriát fogyasztanak**
  - **Csak előrefelé tudnak dolgozni**
  - **Bonyolult az összetett xml-ek megírása**
  - **Az xml-transzformáció borzasztó tud lenni (node csere, node változtatás, etc)**
- XmlDocument, XmlNode, XmlElement, Xml*...
  - **Lassú, sokat fogyaszt (a memóriában felépíti a teljes dokumentum fáját)**
  - **Ezzel sem feltétlenül egyszerűek az algoritmusok**
  - **Előnyös, ha transzformálni akarunk**
- XDocument, XElement, XAttribute, XDeclaration, X*...

XElement

- A konstruktor flexibilis (params kulcsszó) ➔ szinte bármilyen XML létrehozható akár egyetlen konstruktorhívással

- `var xe = new XElement("ember", "Joe");`

  <ember>Joe</ember>

- ```
  var xe2 = new XElement("ember",
      new XElement("név", "Joe"),
      new XElement("kor", 25));
  ```

  <ember>
  <név>Joe</név>
  <kor>25</kor>
  </ember>
  ```

XAttribute

- Konstruktorral:

```
var xe = new XElement("ember",
                      new XAttribute("id", 43984),
            new XElement("név", "Joe"),
            new XElement("kor", 25));
```

- Utólag:

```
var xe2 = new XElement("ember",
            new XElement("név", "Joe"),
            new XElement("kor", 25));
xe2.SetAttributeValue("id", 43984);
```

```
<ember id="43984">
  <név>Joe</név>
  <kor>25</kor>
</ember>
```

Haladó Programozás

LINQ bevezetés, LINQ to Objects
XML kezelés, LINQ to XML
Feladat

XDocument mentése

```
XDocument outDoc = new XDocument(
    new XElement("nép",
        new XElement("ember",
            new XAttribute("id", 0),
            new XElement("név", "Joe"),
            new XElement("kor", 22)),
        new XElement("ember",
            new XAttribute("id", 1),
            new XElement("név", "Quagmire"),
            new XElement("kor", 34))));
outDoc.Save("people.xml");
```

- Betöltés: `XDocument doc = XDocument.Load("http://hp/people.xml");`
- Parse stringből: `XDocument doc = XDocument.Parse(str);`

Egyszerű adatfeldolgozás

```
<emberek>
  <ember id="43984">
    <név>Joe</név>
    <kor>25</kor>
    <telefon>0618515133</telefon>
  <ember>
…
</emberek>
```

.Element(name)
.Attribute(name)

.Elements(name)
.Attributes(name)

```
string nev = xDoc.Element("emberek").Element("ember")
      .Element("név").Value;
int id = int.Parse(xDoc.Element("emberek").Element("ember")
      .Attribute("id").Value);
```

… VAGY … (nem csak attribútum, hanem elem esetén is)

```
int id = (int)(xDoc.Element("emberek").Element("ember")
      .Attribute("id"));
```

LINQ to XML

- X* osztályok: igen erős LINQ-támogatás!
  - **LINQ-zható IEnumerable<T>-ként kapunk vissza egy csomó adatot**

    Pl. XElement xe2:
  - **xe2.Descendants()**
    - **minden gyerekelem (gyerekelemek gyerekelemei is)**
  - **xe2.Descendants("note")**
    - **ilyen nevű gyerekelemek (gyerekelemek gyerekei is)**
  - **xe2.Elements()**
    - **közvetlen gyerekelemek**
  - **xe2.Elements("note")**
    - **ilyen nevű közvetlen gyerekelemek**
  - **xe2.Attributes(), xe2.Ancestors() …**

LINQ to XML

```
<emberek>
  <ember id="43984">
    <név>Joe</név>
    <kor>25</kor>
    <telefon>0618515133</telefon>
  <ember>
…
</emberek>
```

```
XDocument XDoc = ...
var q = from x in XDoc.Root.Descendants("ember")
where x.Element("név").Value.StartsWith("Jo")
select x;
```

XDocument betöltése

- XDocument xDoc = XDocument.Load
  ("people.xml");

```
<person>
        <name>Dr. Kovács Zoltán</name>
        <email>kovacs.zoltan@gmail.com</email>
        <dept> Intézet</dept>
        <rank>igazgató</rank>
        <phone>+36 (1) 666-6666</phone>
        <room>0011</room>
</person>
```

Példa

Listázzuk azokat, akik nem a BA épületben dolgoznak

```
XDocument XDoc = XDocument.Load( "http://users.nik.uni-obuda.hu/hp/people.xml");
var q0 = from akt in XDoc.Descendants("person")
            let room=akt.Element("room").Value
            where !room.StartsWith("BA")
            select akt;
foreach (var akt in q0) {
        Console.WriteLine(akt.ToString());
}
```

Feladatok

Importáljuk az XML-t objektumlistába (készítsünk saját osztályt az adatok tárolásához)

1. Határozzuk meg az All intézetben dolgozók darabszámát!
2. Az All dolgozóit listázzuk „lapozva”, 15 elemenként kelljen ENTER-t leütni
3. Jelenítsük meg azokat, akiknek a harmadik emeleten van irodája!
4. Kiknek van a leghosszabb vagy legrövidebb nevük?
5. Határozzuk meg intézetenként a dolgozók darabszámát!
6. Határozzuk meg a legnagyobb intézetet!
7. Listázzuk a legnagyobb intézet dolgozóit!
8. Listázzuk a harmadik legnagyobb intézet dolgozóit szobaszám szerint csökkenő sorrendben!

Összefoglalás

- LINQ to Object
- Bővítő metódusok
- XML
- LINQ to XML

Feladat - Értelmezze a leírást

- public static System.Collections.Generic.IEnumerable<TSource> xxxmetod<TSource> (this System.Collections.Generic.IEnumerable<TSource> source, TSource element);
- public static int? Max (this System.Collections.Generic.IEnumerable<int?> source);
- public static System.Linq.IOrderedEnumerable<TSource> ThenBy<TSource,TKey> (this System.Linq.IOrderedEnumerable<TSource> source, Func<TSource,TKey> keySelector);