Typy ogólne i kolekcje

Metoda ogólna

```
class Klasa {
  public static void Swap<T>(ref T a, ref T b) {
      T tmp = a;
      a = b;
      b = tmp;
  }
  public static T Max<T>(T a, T b)
          where T: IComparable
  {
   if(a.CompareTo(b) > 0)
      return a;
   return b;
  }
}
```

Klasa ogólna

```
class Nazwa<T,K>
      where T: IComparable<T>
      where K: BaseClass, ICloneable{
}
```

```
class Lista<T>
{
   class Node {
      public T Data;
      public Node Next;
   }
   Node head;
   ...
}
```

Ograniczenia typu *parametrycznego*

- where T:
  - lista interfejsów
  - klasa bazowa (tylko jedna)
  - struct - T jest typem wartości
  - class - T jest typem referencyjnym
  - new() - T musi posiadać publiczny bezparametrowy konstruktor
- Wartość domyślna ("zerowa")
  - T temp = default(T);

Interefejs IEnumerable i IEnumerator

- IEnumerable
  - **IEnumerator GetEnumerator()**
- IEnumerable<T>
  - **IEnumerator<T> GetEnumerator()**
- IEnumerator
  - **bool MoveNext()**
  - **void Reset()**
  - **Object Current { get; }**
- IEnumerator<T>
  - **T Current { get; }**

Porównanie obiektów

- IComparable<T>
  - **int CompareTo(T other)**
- IComparer<T>
  - **int Compare(T x, T y)**

Kolekcje

```
List<string> lista = new List<string>()
{ "jabłko", "banan", "kiwi" };

lista.Add("gruszka");
string owoc = lista[3];
foreach(string x in lista){
    Console.WriteLine(x);
}
```

Słowniki generyczne

```
Dictionary<string, int> slownik =
          new Dictionary<string, int>();

slownik.Add("Goblins", 1992);
slownik.Add("Mortal Kombat", 1993);
slownik.Add("Defender Of Crown", 1988);

int rok = slownik["Mortal Kombat"];

foreach (KeyValuePair<string, int> para in slownik) {
   Console.WriteLine(para.Key);
   Console.WriteLine(para.Value);
}
```

LINQ

- LINQ: Language Integrated Query
- Alternatywa dla SQL
- Weryfikacja podczas kompilacji kodu
- W ten sam sposób możemy *pytać*: kolekcje, XML oraz bazy danych
- Przykłady:
  - https://code.msdn.microsoft.com/101-LINQ-Samples-3fb9811b