Kollekciók, Generics

Csorba Kristóf

IEnumerable

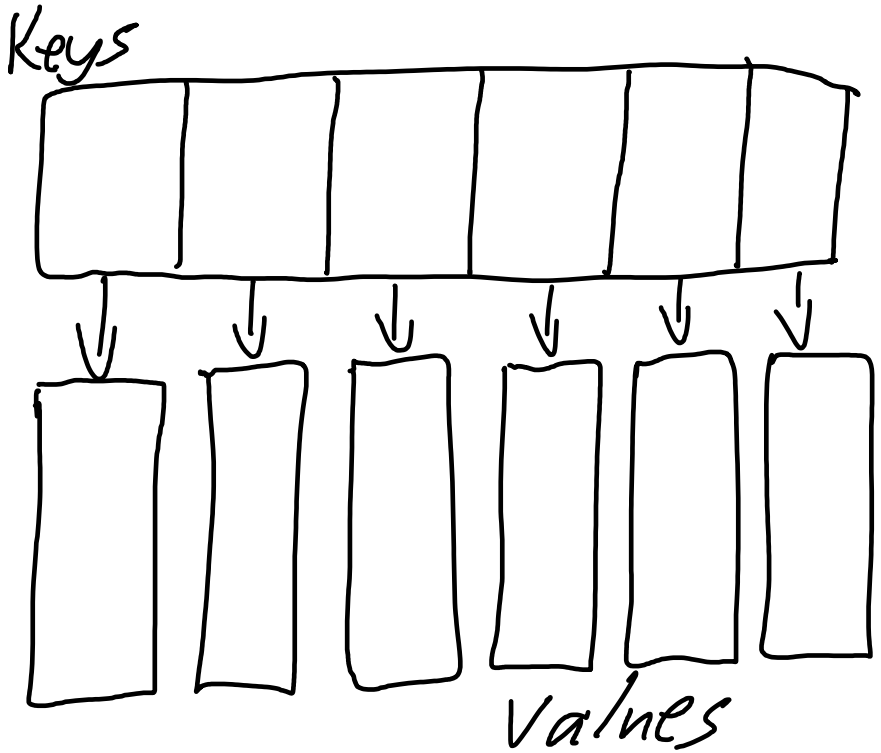
- Felsorolható…
  - List
  - tömbök
  - string (karaktereket kapunk)
  - …

Generikus tárolók (ism.?)

(Java alatt is vannak.)

Dictionary, generics

```
public readonly Dictionary<XElement, AnimationCommand[]> Commands;
```



e:\Projektek\evipdev\svg2pptx\svg2pptx\SvgWrapper.cs

Enum és Dictionary init

```
// EVIP: initializing a dictionary with values
readonly Dictionary<InitialShapeEnum, SPoint[]> initialShapes =
    new Dictionary<InitialShapeEnum, SPoint[]>()
    {
        { InitialShapeEnum.Rectangle, new SPoint[] {
            new SPoint(50,50,0.0),
            new SPoint(450,50,0.25),
            new SPoint(450,450,0.5),
            new SPoint(50,450,0.75),
            new SPoint(50,50,1.0)
            }
        },
        { InitialShapeEnum.Triangle, new SPoint[] {
            new SPoint(50,50,0.0),
            new SPoint(450,50,0.33),
            new SPoint(50,450,0.67),
            new SPoint(50,50,1.0)
            }
        }
    };
```

```
// EVIP: enum
public enum InitialShapeEnum
{
    Rectangle, Triangle
}
```

foreach, yield return

```
static void Main(string[] args)
{
    foreach(var j in GetNumbers())
        Console.WriteLine(j);
}

static IEnumerable<int> GetNumbers()
{
    for (int i = 0; i < 5; i++)
        yield return i;
}
```

foreach, yield return

```
// EVIP: Method with yield return
public override IEnumerable<SPoint> Transform(IEnumerable<SPoint> points)
{
    var controlPoints = points.ToList();
    for (int i = 0; i < controlPoints.Count - 1; i++)
    {
        bool isLast = (i == controlPoints.Count - 2);
        var newPoints = GetPointsBetween(
            controlPoints[i], controlPoints[i + 1], isLast);
        foreach (var p in newPoints)
            yield return p;
    }
}
```

yield return, Range

```
// EVIP: yield return to create IEnumerable
// EVIP: Enumerable.Range
public static IEnumerable<Point> GetAllPixelLocations(int width, int height)
{
    foreach (int y in Enumerable.Range(0, height - 1))
        foreach (int x in Enumerable.Range(0, width - 1))
            yield return new Point(x, y);
}

// EVIP: alternative parameter list for a polymorph method to make calling side
//  more compact and readable.
public static IEnumerable<Point> GetAllPixelLocations(this WriteableBitmap image)
{
    return GetAllPixelLocations(image.PixelWidth, image.PixelHeight);
}
```

Mandelbrot\MandelbrotCommon\ImageIndexingExtensions.cs

yield return

```
// EVIP: IEnumerable and foreach with yield return
public override IEnumerable<SPoint> Transform(IEnumerable<SPoint> points)
{
    // 1 is the full length of the line in terms of t parameter.
    // 'periodCountInOneIteration' rotation periods are preformed
    //      during one [0;1] iteration.
    double periodCountInOneIteration = (double)FullRotationCountDuringSingleIteration
        + 1.0 / (double)NumberOfDifferentIterations;
    double angularSpeed = 2.0 * Math.PI * periodCountInOneIteration;
    foreach (var p in points)
    {
        // Starting from up and turning clockwise.
        p.X += Radius * Math.Sin(p.T * angularSpeed);
        p.Y += -Radius * Math.Cos(p.T * angularSpeed);
        yield return new SPoint(p);
    }
}
```

SpirographLab\SpirographLib\Transformations\Rotation.cs

foreach és Dictionary.Keys

```
public void ApplyCommandsForFrameIndex(int frameIndex)
{
    // EVIP: go along key collection of dictionary
    foreach (var element in Commands.Keys)
    {
```

```
public class SvgWrapper
{
    private readonly XElement root;
    public readonly Dictionary<XElement, AnimationCommand[]> Commands;
```

svg2pptx\svg2pptx\SvgWrapper.cs

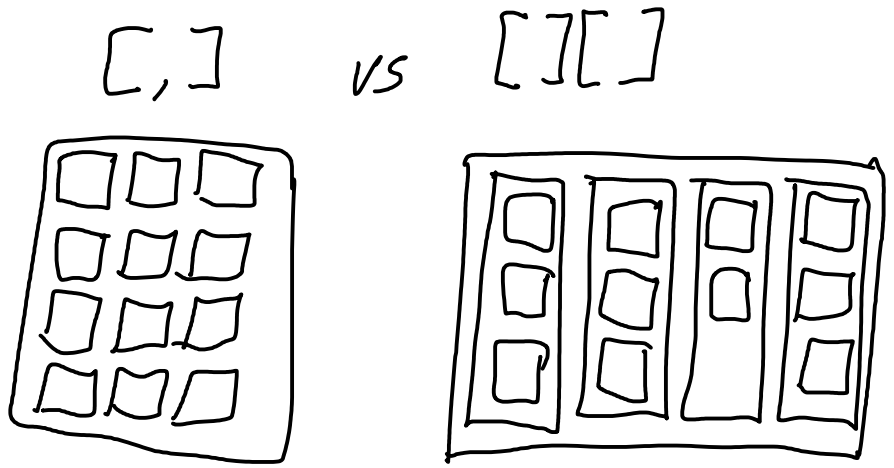
IEnumerable és tuple

```
public static void SetPixels(this IEnumerable<(Point, Color)> values,
    WriteableBitmap image)
{
    // EVIP: using context of WritableBitmap.
    //  (Otherwise, it will be extremely slow.)
    // EVIP: using and IDisposable
    using (image.GetBitmapContext())
    {
        // EVIP: Tuple and deconstruction
        foreach ((Point p, Color c) in values)
            image.SetPixel((int)p.X, (int)p.Y, c);
    }
}
```

Mandelbrot\MandelbrotCommon\ImageIndexingExtensions.cs

2D array: [][] vs [,]

```
// EVIP: instantiating and initializing 2D array (row, column)
Fields = new Field[size, size];
for (int row = 0; row < size; row++)
    for (int col = 0; col < size; col++)
        Fields[row, col] = new Field()
            { Row = row, Column = col, Owner = 0 };
```



AttaxxPlus\AttaxxPlus\Model\SimpleGame.cs

2D array mérete

```
// EVIP: getting sizes of two dimensional array
for (int row = 0; row < Model.Fields.GetLength(0); row++)
{
```

```
public Field[,] Fields { get; protected set; }
```

AttaxxPlus\AttaxxPlus\ViewModel\GameViewModel.cs

```
// EVIP: static factory method (using private static helper method as well)
public static IEnumerable<AnimationCommand> Parse(string s)
{
    // EVIP: Regex to extract values from a string
    Regex r = new Regex(@"([a-z])([0-9]*):([0-9]*)(-([0-9]*))?");
    var matches = r.Matches(s);
    foreach(Match m in matches)
    {
        // Note: group 0 is the entire match. Group 1 is the letter in the first b
        string type = m.Groups[1].Captures[0].Value;
        int? param = GetIntFromRegexpMatch(m, 2);
        int? firstFrame = GetIntFromRegexpMatch(m, 3);
        int? lastFrame = GetIntFromRegexpMatch(m, 5);


        // If there is no "-" sign, firstFrame is the same as lastFrame ("v:2" app
        bool hadIntervallSign = m.Groups[4].Captures.Count > 0;
        if (lastFrame == null && !hadIntervallSign)
            lastFrame = firstFrame;

        switch(type)
        {
            case "v":
                yield return new VisibleCommand(param, firstFrame, lastFrame);
                break;
            case "e":
                yield return new EmphasizedCommand(param, firstFrame, lastFrame);
                break;
```

“Jó kérdés…”

Paraméterként átadva **IEnumerable<>** vagy **List<>** a jobb? Mikor melyik?

Jövő héten HF1a code review

...aminek csak akkor van értelme, ha már van valami készen.

Megjegyzések:

- char[]
- érték típusú elemekkel tömb
- string <-> char[] (ToCharArray())
- indexelése, értékadások