Exercises

September 26th 2017

This exercise sheet can be downloaded as PDF

Exercise 1 - Explain the code

Given below code

```
public class Application
{

    public delegate double NumericFunction(double d);
    static double factor = 4.0;

    public static NumericFunction MakeMultiplier(double factor)
    {
        return delegate (double input) { return input * factor; };
    }

    public static void ApplicationMain()
    {
        NumericFunction f = MakeMultiplier(3.0);
        double input = 5.0;

        Console.WriteLine("factor = {0}", factor);
        Console.WriteLine("input = {0}", input);
        Console.WriteLine("f is a generated function which multiplies its
input with factor");
        Console.WriteLine("f(input) = input * factor = {0}", f(input));
    }
}
```

When running it we get the following output:

```
factor = 4
input = 5
f is a generated function which multiplies its input with factor
f(input) = input * factor = 15
```

Explain the output without running the code.

- Why is factor 4?
- Why is the result of the operation *15*

Exercise 2 - Shuffle a list

Write a shuffle operation that shuffles a given collection of elements in a random fashion. There exists no shuffle operation in the .Net libraries.

You decide on programming either a mutating or a non-mutating variant of the operation. Be sure to understand the difference between these two options.

You might want to consider using generics to allow the shuffle operation to work on all collections.

Exercise 3 -Explicit use of an iterator

Modify the code below to use an iterator(enumerator) instead of a foreach-loop

```
public class IteratorExercise
{
    private IList<string> collectionOfStringList;

    public IteratorExercise()
    {
        collectionOfStringList = new List<string>
        {
            "Hello", "my", "dear", "friend.", "How", "are", "you", "today?"
        };
    }

    public void PrintSentenceToConsole()
    {
        foreach (var word in collectionOfStringList)
        {
            Console.Write(word);
            Console.Write(" ");
        }
        Console.WriteLine("");
    }
}
```

Hint: GetEnumerator() and while-loop is your friend.

Exercise 4 - Multiple iterators

Firstly download the following files:

- Interval.cs
- Pair.cs

The interval class defines a range. It has an implemented Enumerator. The Pair class is a helper to make it easier to print the pairs.

For a given interval *I*, generate a list of all possible pairs *(e,f)* where both *e* and *f* comes from *I*.

As an example, observer the Interval(1,2) should give the following pairs:

- (1,1)
- (1,2)
- (2,1)
- (2,2)

For the purpose of this exercise, request to enumerators from the Interval and traverse the interval using two nested while-loops.

It is required for you to use Enumerators and not foreach-loops.

Exercise 5 - Research the *yield*-keyword

Find out what the *yield*-keyword does and how to use it.