Expression Tree

Выполните оба приведенных ниже задания. Шаблоны доступны по линке.

Задание 1 – Expression Transformation

(ExpressionTrees.Task1.ExpressionsTransformer.csproj).

Создайте класс-трансформатор на основе ExpressionVisitor, выполняющий следующие 2 вида преобразований дерева выражений:

- Замену выражений вида <переменная> + 1 / <переменная> - 1 на операции инкремента и декремента
- Замену параметров, входящих в lambda-выражение, на константы (в качестве параметров такого преобразования передавать:
    - Исходное выражение
    - Список пар <имя параметра: значение для замены>

Для контроля полученное дерево выводить в консоль или смотреть результат под отладчиком.

Здесь можно использовать ExpressionTreeVisualizer или другой визуализатор, или вовсе обойтись без помощи визуализатора.

Задание 2 - Mappers

(ExpressionTrees.Task2.ExpressionMapping.csproj, ExpressionTrees.Task2.ExpressionMapping.Tests.csproj).

Используя возможность конструировать ExpressionTree и выполнять его код, создайте собственный механизм маппинга (копирующего поля (свойства) одного класса в другой).

Приблизительный интерфейс и пример использования приведен ниже (MapperGenerator – фабрика мапперов, Mapper – класс маппинга).

Обратите внимание, что в данном примере создается только новый экземпляр конечного класса, но сами данные не копируются. Задача - доработать маппер так, чтобы данные полей, совпадающих по имени и типу, копировались из одного класса в другой.

При желании можно предусмотреть некоторые простые преобразования из одного типа в другой.

```
public class Mapper<TSource, TDestination>
{
    Func<TSource, TDestination> mapFunction;
    internal Mapper(Func<TSource, TDestination> func)
    {
        mapFunction = func;
    }
    public TDestination Map(TSource source)
    {
        return mapFunction(source);
    }
}
```

```
public class MappingGenerator
{
    public Mapper<TSource, TDestination> Generate<TSource, TDestination>()
    {
        var sourceParam = Expression.Parameter(typeof(TSource));
        var mapFunction =
            Expression.Lambda<Func<TSource, TDestination>>(
            Expression.New(typeof(TDestination)),
            sourceParam
            );


        return new Mapper<TSource, TDestination>(mapFunction.Compile());
    }
}
```

```
public class Foo { }
public class Bar { }


[TestMethod]
public void TestMethod3()
{
    var mapGenerator = new MappingGenerator();
    var mapper = mapGenerator.Generate<Foo, Bar>();


    var res = mapper.Map(new Foo());
}
```