intech
ECOLE SUPERIEURE
D'INGENIERIE INFORMATIQUE

Exercice : No Monkey Tree

invenietis

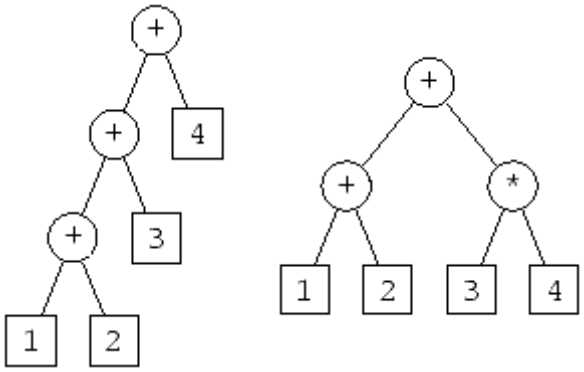
1 Objectif

Ce TP a pour but de vous introduire la notion d'AST (**A**bstract **S**yntax **T**ree) à travers la manipulation de **Linq.Expressions** qui propose un AST prêt à l'emploi. L'AST est un concept fondamental lorsque l'on veut notamment s'aventurer vers des thèmes comme la théorie de la compilation.

2 Sharpen your saw

2.1 AST

Un **A**bstract **S**yntax **T**ree est une représentation du code sous la forme d'une structure d'arbre. Chaque nœud de l'arbre représente une instruction. L'ensemble d'un programme peut être représenté sous la forme d'un AST. Voici un exemple concret :



Ici on peut voir 2 AST simples :

- Le premier représente simplement : 1 + 2 + 3 +4
- Le second : 1 + 2 + 3 * 4

Dans le processus de compilation du code source, un AST est généré, puis ce dernier est exécuté. Il est très bien possible de retrouver le même AST depuis un code à la base en C#... ou en Javascript !

2.2 Linq.Expressions

Linq.Expressions est le namespace qui contient tout ce qu'il faut afin de nous permettre de manipuler des objets prenant la forme d'arborescences d'expression (Expression Tree). Une arborescence d'expression représente un AST d'où le fait qu'il nous évite la tâche ardue d'en implémenter un nous-même.

https://msdn.microsoft.com/en-us/library/system.linq.expressions(v=vs.110).aspx

La classe abstraite *Expression* permet de modéliser des arborescences d'expression. *Expression* est la classe root afin de construire des nœuds d'arborescence d'expression:

https://msdn.microsoft.com/en-us/library/system.linq.expressions.expression(v=vs.110).aspx

Expression Tree, c'est quoi ?!

Une arborescence d'expression permet de traduire du code exécutable en données (data). On peut ainsi avoir une structure de données représentant l'ensemble de notre programme. Il est ainsi possible d'analyser du code exécutable, de le modifier ou encore de le transformer avant exécution ! À quoi ça peut servir ? Pensez LINQ to SQL...

Je vous invite fortement à lire cet article de Charlie Clavert afin de mieux visualiser l'utilisation et la puissance des arborescences d'expression :

https://blogs.msdn.microsoft.com/charlie/2008/01/31/expression-tree-basics/

2.3 Reverse Polish Notation

Un test vous introduira au RPN dans lequel vous allez devoir construire la notation RPN depuis une arborescence d'expression. Dans notre cas, une opération "5 * (3 + 2)" deviendra "5 3 2 + x". Il existe des calculatrices RPN encore utilisées de nos jours et comme vous avez peut-être pu le remarquer, la notation ressemble beaucoup à une structure de données que vous êtes censés connaître.

Il vous faudra utiliser le pattern du visiteur en implémentant correctement la classe *VisitorReversePolishNotation* (Voir 2.4).

Pour en savoir plus sur le RPN:

https://en.wikipedia.org/wiki/Reverse_Polish_notation

2.4 Visitor

Le pattern du visiteur est un concept fondamental lorsque l'on veut parcourir et même manipuler une arborescence d'expression à travers des mutations par exemple.

Documentation concernant la classe *ExpressionVisitor*:

https://msdn.microsoft.com/fr-fr/library/system.linq.expressions.expressionvisitor(v=vs.95).aspx

3 À vous de jouer !

Les premiers tests vous permettent de vous familiariser avec les factory méthodes de Expression avant d'entamer des opérations un peu plus complexes comme l'utilisation du design pattern du visiteur, incontournable lorsque l'on veut travailler sérieusement avec les arborescences d'expression.

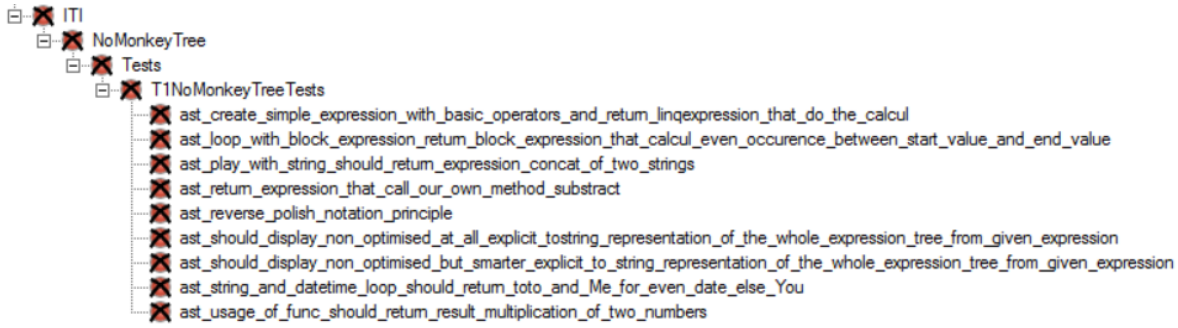
Les tests peuvent être réalisés dans n'importe quel ordre, mais nous vous conseillons de les implémenter au fur et à mesure car les tests sont de difficultés croissantes, ce qui ne devrait pas être le cas mais nous sommes dans le cadre d'un TP.

Vous disposez d'une solution comprenant 2 projets :

- ITI.NoMonkeyTree.Tests, contient les tests unitaires
- ITI.NoMonkeyTree, contient l'implémentation de l'AST que vous allez devoir compléter.

Pour faire tourner les tests unitaires il vous suffit de configurer le projet ITI.NoMonkeyTree.Tests en tant que projet de démarrage puis d'exécuter la solution.

Comme vous pouvez le constater, pour le moment, tous les tests sont rouges :



À vous de faire en sorte qu'ils passent en vert. Pour cela, vous avez le droit de faire ce que bon vous semble dans le projet ITI.NoMonkeyTree mais il vous est bien évidemment interdit de modifier le projet de test pour valider les diodes…

Les tests unitaires sont là pour spécifier de façon détaillée les fonctionnalités attendues.

Pour aller plus loin :
http://referencesource.microsoft.com/#System.Core/Microsoft/Scripting/Ast/Expression.cs