Mission 2 : Rapport Final

Groupe 2.2 - De Grove Gil, Lemaire Jérôme

Vendredi 17 octobre 2014

Introduction

Dans cette mission, notre but était d'implémenter un programme permettant de dériver des expressions arithmétiques de manière automatisé. Pour ce faire, il nous était demandé d'implémenté une solution utilisant un arbre.

Description de l'implémentation

Chaque expression arithmétique est tout d'abord décomposée en plusieurs opérations, pour ce faire nous avons utilisé la classe OperationNode qui est étendue en plusieurs types de noeuds appelés selon l'operation représentée(AddNode, SubNode, etc.).
Cette façon de faire, nous permet de simplifier la gestion des dérivées. En effet, une fois l'arbre arithmétique construit, il nous suffit de dériver celui-ci pour obtenir le résultat désiré.

Pourquoi plusieurs types de noeuds ?

L'utilisation de plusieurs types de noeuds etait pour nous une évidence a cause du contexte de la mission. Chaque opération se dérive de manière unique et par conséquence la gestion par l'utilisation de classes distinctes représente au mieux la réalité.
De plus cela facilite l'ajout de nouvelle opération, qui ne sont pas implémentée en ce moment.

Illustration d'une expression.

Comme expliqué au point précédent, nous allons maintenant illustré la décomposition d'une expresison en opération primaire dans notre programme. Dans la 1, nous voyons qu'il y a une hiérarchie dans la manière dont nous utilisons les noeuds.

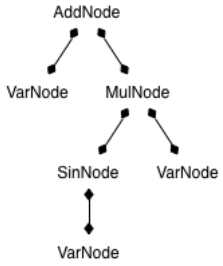

FIGURE 1 – Illustration de l'expression $(x + x * sin(x))$

Diagramme de classe.

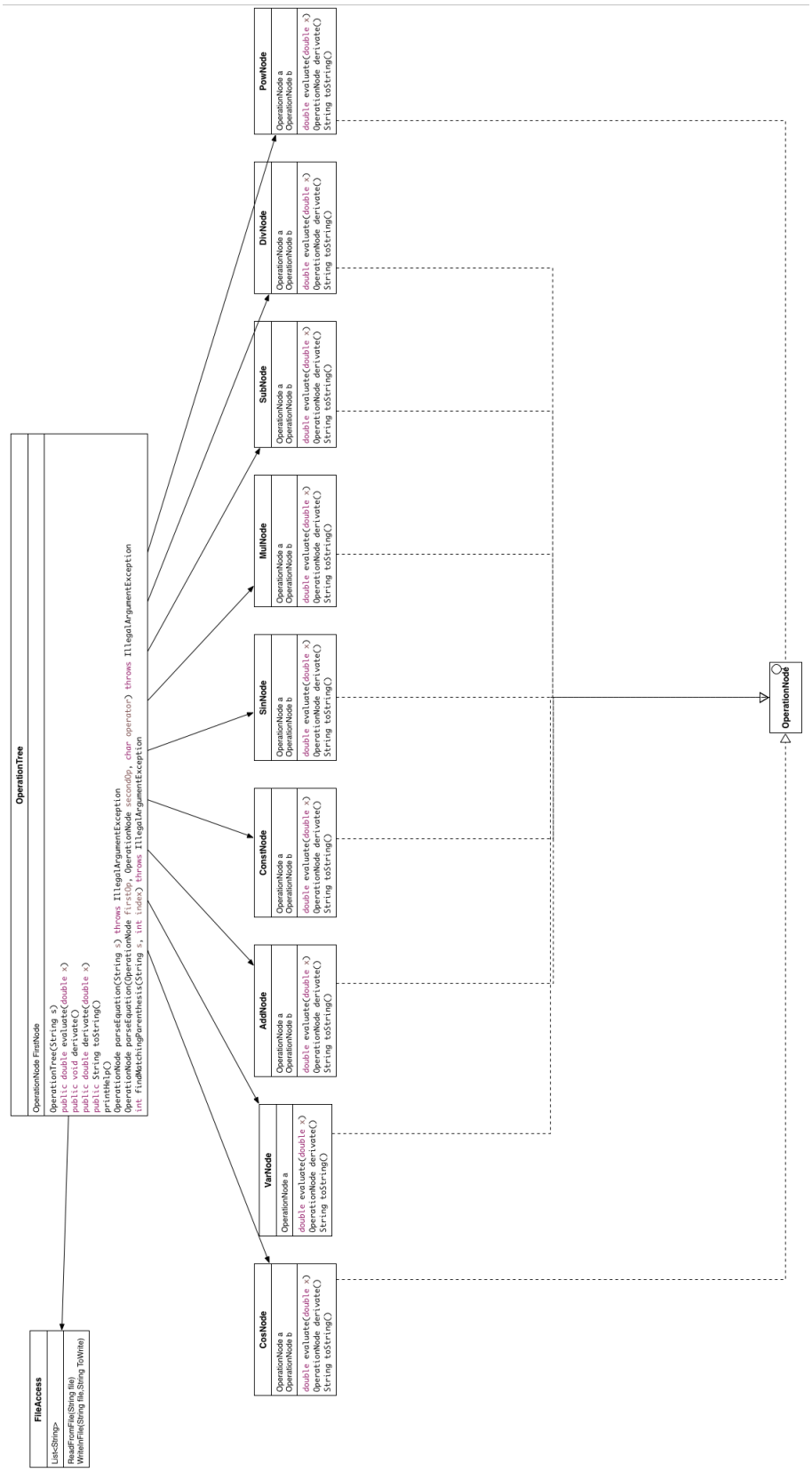

FIGURE 2 – Diagramme de classe de notre programme.

Come expliqué dans les points précédents, la figure2 représente les différentes classes utilisées. Nous voyons bien que la classe principale (OperationTree) a accès a n'importe quel noeud. Cependant, dans ses attributs, nous avons mis un OperationNode.
La classe FileAccess nous permet l'accès aux fichiers. Elle provient de la mission précédente.

Problèmes rencontrés

— Dans notre implémentation, nous avons eu un problème pour créer une structure doublement chainée. Pour descendre dans l'arbre, nous pouvons le faire simplement. Maintenant pour remonter d'un enfant vers un parent le lien est inexistant. Cette implémentation avec un lien vers le parent, serait utile pour la simplification des dérivées(présence de 0 dans l'expression).

— Tout d'abord, même si cela n'est pas une excuse, le manque d'organisation dans le groupe n'a pas facilité la réalisation de la tâche. Nous nous sommes rendus compte que le travail était mal répartit. Nous mettrons donc l'organisation des taches au centre de nos prochaines réunions pour les missions.

Conclusion

La décomposition des classes dans notre programme, aurait du faciliter la répartition des taches, l'utilisation de multiple classe permet de pouvoir être plus modulable. Tant au niveau de la repartition du travail, qu'au niveau de l'ajout de nouvelles fonctions pour adapter notre programme.