

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'opération 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 était pour nous une évidence a cause du contexte de la mission. Chaque opération se dérive de manière unique et par conséquent 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.

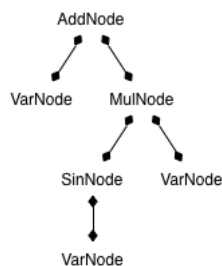


FIGURE 1 – Illustration de l'expression $(x + x * \sin(x))$

Diagramme de classe.

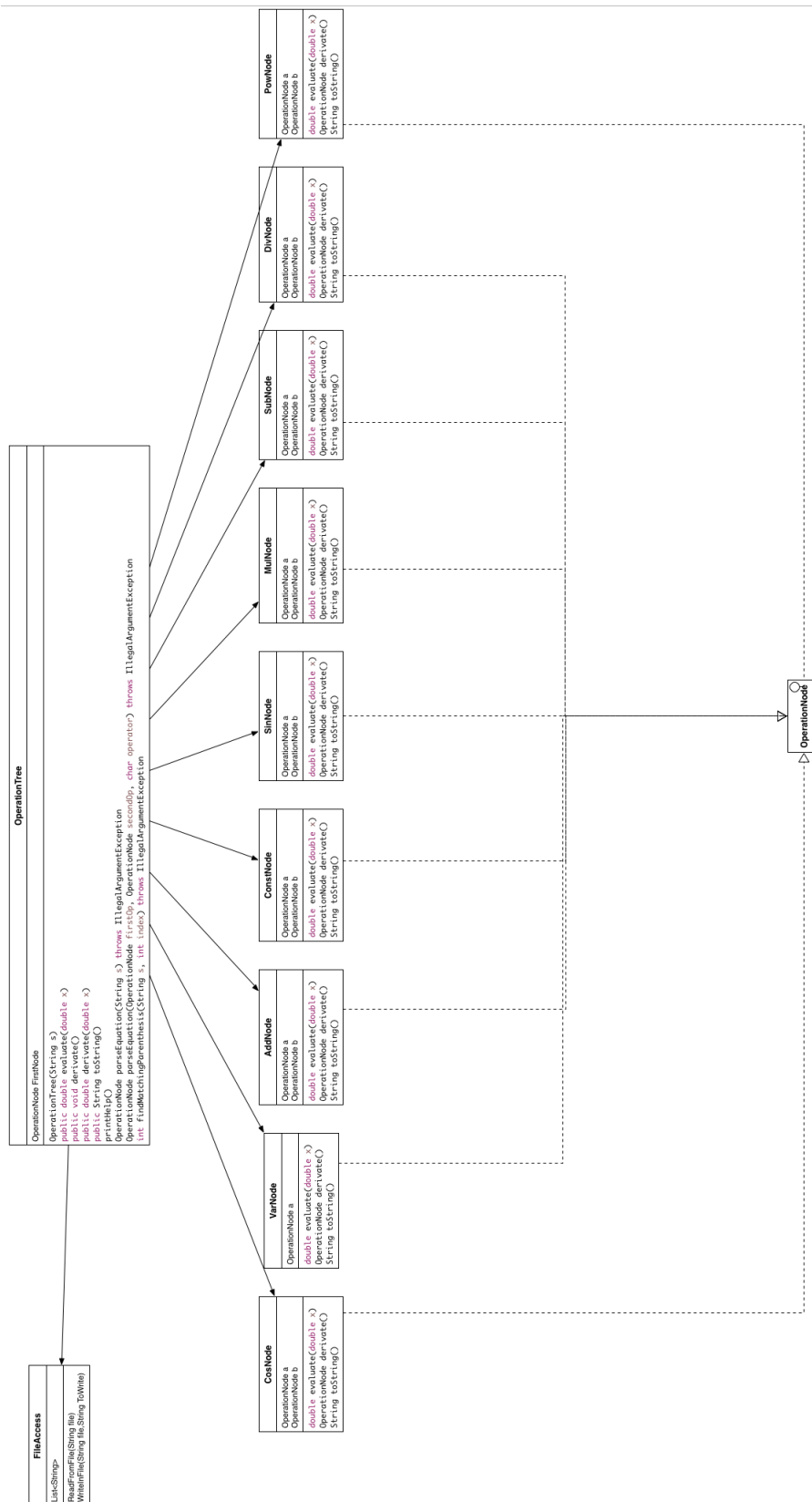


FIGURE 2 – Diagramme de classe de notre programme.

Problèmes rencontrés

Conclusion