

# Algorithmique et structures de données : Mission 2

Groupe 1.2: Ivan Ahad - Jérôme Bertaux - Rodolphe Cambier  
Baptiste Degryse - Wojciech Grynczel - Charles Jaquet

18 octobre 2014

Rapport écrit par Ivan Ahad et Rodolphe Cambier

## Introduction

L'objectif de cette mission est de créer un programme qui prend en entrée une expression analytique et qui retourne la dérivée de celle-ci, grâce à l'utilisation d'arbres binaires.

## Fonctionnement

Le programme lit le fichier avec les expressions à résoudre et convertit les lignes du fichier en un `ArrayList`:

```
1 ArrayList<String> expressions = ReadWrite.mRead("expression.txt");
```

Il traite ensuite chaque élément de l'`ArrayList` indépendamment:

```
1 for (String expression : expressions)
```

Pour chaque expression (élément de l'`ArrayList`), il va transformer la ligne de texte en un `FormalExpressionTree`, un arbre binaire qui va contenir les expressions afin qu'elles soient traitables par la suite. Cela sera réalisé lors de la création de l'arbre, quand le constructeur de *FormalExpressionTreeImpl* appelle la fonction *parse*.

```
1 FormalExpressionTree fet = new FormalExpressionTreeImpl(expression);
```

Une fois transformées en arbre, les expressions mathématiques vont être dérivées en appelant la fonction *derive*, de la classe *Calculator* sur l'arbre représentant l'expression mathématique à dériver. Cette fonction va traiter l'élément du root, et va s'appeler récursivement sur chacun des enfants du root, de sorte que tout l'arbre soit traité. A chaque étape, il remplit un nouvel arbre, avec la dérivée de l'expression de départ qu'il retourne finalement.

```
1 public static LinkedBinaryTree<String> derive(LinkedBinaryTree<String> t){
2     LinkedBinaryTree<String> t2=null;
3     if(t.element().equals("+"))
4         t2= new LinkedBinaryTree<String>("+", derive(t.leftTree()), derive(t.rightTree()));
5     else if(t.element().equals("-"))
6         t2= new LinkedBinaryTree<String>("-", derive(t.leftTree()), derive(t.rightTree()));
7
8     [...]
9
10
11     else
12         System.out.println("element inconnu : "+ t.element());
13     return t2;
14 }
```

Finalement, l'arbre de départ ainsi que le nouvel arbre représentant sa dérivée seront imprimés.

**Tests réalisés**

**UML**