# Projet Java, L3 Miage, 2014-2015 :

## The fork interpreter

### Contraintes :

- A rendre pour le 7 janvier 2015 (délai de rigueur)
- A réaliser par groupe de trois personnes (à déposer sur arche).

### Description du sujet :

Ecrire un interpréteur en Java pour un mini langage impératif. L'interpréteur devra exécuter sur une invite de commande les commandes du langage défini par la grammaire suivante :

```
Com ::= Ide := NExp                        assignation
      | Com ; Com                          séquence
      | if BExp then Com else Com          conditionelle
      | while BExp do Com                  iteration
      | let Ide in Com end                 declaration de variable
      | let Ide be Ide in Com end          aliasing
      | Ide := fork()                      process creation
      | return NExp                        return statement
      | Ide := wait(NExp)                  wait
```

où `Nexp` représente les expressions numériques :

```
NExp ::=  Num | Ide | NExp NOp NExp
NOp  ::=  + | * | - | /
Num  ::= 0 | 1 | 2 | ...
```

et `Bexp` représente les expressions booléennes :

```
BExp ::= true | false | NExp COp NExp | not BExp | BExp BOp BExp
COP  ::= < | =
BOP  ::= and | or
```

Les variables de `Ide` sont générées comme étant des séquences de une ou plusieurs lettres (les autres caractères sont proscrits).
La déclaration de variable permet de définir une variable locale (n'a de valeur que dans la commande) tandis que l'aliasing permet de dire que les deux noms de variables passés en paramètre sont les mêmes.

### Consignes (étapes) :

1. Ecrire un programme permettant de manipuler des chaînes de caractères données en entrée, faites en sorte qu'il termine lorsque la chaîne de caractères saisie par l'utilisateur a la valeur "EXIT".

2. Pour chaque commande entrée en paramètre, générer son arbre syntaxique (cherchez référence Abstract Syntax Tree –AST - sous Google ou wikipedia) et générer une exception en

cas d'erreur (il peut y avoir différents types d'erreur : syntaxique, conflit de noms, …). A cet effet, il faudra choisir une modélisation appropriée des AST en Java.

3. Exécuter la commande (si c'est possible). A cet effet, vous devrez définir et modifier un environnement, associant une valeur (entière ou booléenne) aux variables déclarées.

4. Gérer les commandes fork et wait en utilisant des threads

5. Programmer une petite interface graphique de saisie des commandes, permettant aussi d'importer des commandes enregistrées sur un fichier.

6. Prévoir une option de sauvegarde d'une commande. Et une option de sauvegarde d'un environnement en utilisant streams et sérialisation.

Vous devrez rendre :

- une archive .zip contenant votre programme compilé avec ses sources (.class, .java).
- et un rapport au format pdf (de 10 pages maximum strict !) expliquant vos choix, les difficultés rencontrées.

## Evaluation :

L'évaluation portera sur :

1. Une soutenance orale où vous ferez une démonstration de votre programme (5 pt) (le 8 janvier)

2. La correction du programme et le respect des consignes (10 pt).

3. La clarté et la pertinence du rapport (5 pt)