TP Analyse lexicale.

L'objectif de cette séance est de réaliser un analyseur lexical[1] pour une partie du langage Java[2].
La partie lexicale du langage est spécifiée par le document :
`https://docs.oracle.com/javase/specs/jls/se11/html/jls-3.html`.
Nous utiliserons le langage OCAML et plus precisément l'outil `camllex`.

1 Analyseur lexical

Un analyseur lexical est la première étape de l'analyse d'un langage. Celle-ci est suivie par l'analyse syntaxique[3] qui sera l'objet des futures séances de TPs et l'analyse sémantique[4] qui sera l'objet de l'UE Sémantique et Traduction des Langages de la majeure Sciences et Ingénierie du Logiciel, et de la matière Traduction des Langages des majeures High Performance Computing et Big Data et Images et MultiMedia.

Un analyseur lexical est spécifié sous la forme d'une séquence d'association entre une expression régulière et une action. L'action est effectuée lorsque l'analyseur lexical reconnait un mot qui fait partie du langage décrit par l'expression régulière associée. Si plusieurs expressions régulières reconnaissent le même mot, seule l'action associée à la première expression régulière est exécutée. Si plusieurs mots reconnus par des expressions régulières contiennent un même préfixe, alors l'action exécutée est celle associée à l'expression régulière qui reconnait le mot le plus long.

2 L'outil camllex

Pour réaliser cette séance, nous allons utiliser l'outil `camllex`[5] qui traduit la spécification d'un analyseur lexical en un programme `caml`[6] qui implante un analyseur lexical sous la forme d'un automate fini.

3 Etude d'un exemple

L'objectif est d'étudier l'exemple suivant en s'appuyant sur la documentation de l'outil `camllex` :
`http://caml.inria.fr/pub/docs/manual-ocaml/lexyacc.html`.
L'analyseur lexical est défini dans le fichier `lexerJava.mll`.
Le programme principal qui utilise l'analyseur lexical généré est défini dans le fichier `mainJava.ml`.
Les unités lexicales reconnues par l'analyseur lexical et affichées par le programme principal au fur et mesure de la lecture d'un fichier sont définies dans le fichier `tokenJava.mli`.
Pour compiler le programme, utilisez la commande `dune build mainJava.exe` qui produit l'exécutable `mainJava.exe` dans le répertoire habituel `_build/default`. Cet exécutable prend en paramètre le nom du fichier que l'on veut analyser.

4 Construction d'un analyseur lexical

L'objectif de cet exercice est de compléter le fichier fourni en exemple `lexerJava.mll` pour traiter l'ensemble des unités lexicales de Java telles qu'elles sont décrites dans la documentation :
`https://docs.oracle.com/javase/specs/jls/se11/html/jls-3.html`.
Les unités lexicales à définir sont listées dans le fichier `tokenJava.ml` qui contient également la fonction `printToken` utilisée par le programme principal pour afficher les résultats de l'analyse lexicale.

[1] `http://en.wikipedia.org/wiki/Lexical_analysis`
[2] `http://en.wikipedia.org/wiki/Java_(programming_language)`
[3] `http://en.wikipedia.org/wiki/Parsing`
[4] `http://en.wikipedia.org/wiki/Semantics`
[5] `http://caml.inria.fr/pub/docs/manual-ocaml/lexyacc.html`
[6] `http://caml.inria.fr/pub/docs/manual-ocaml/`

Le fichier `lexerJava.mll` contient déjà la plupart des unités lexicales simples. **Vous vous attacherez à traiter les différentes formes de constantes (entiers, réels, caractères, chaînes de caractères sous toutes les formes possibles) ainsi que les différentes formes de commentaires (ligne ou bloc).**