NF11 – TP2 : GENERATION D'ANALYSEUR LEXICAL ET SYNTAXIQUE (1ERE SEANCE)

GENERALITES

OBJECTIFS

Créer une grammaire du langage Logo et une représentation intermédiaire d'un programme Logo.

Générer un analyseur syntaxique de cette grammaire.

Créer un interpréteur graphique du langage Logo.

Crée différentes tables d'un programme.

Vérifier quelques conditions sémantiques.

OUTILS

Générateur d'analyseur lexical et syntaxique (lexer – parser) à partir de grammaires AntLR Version 4.

Matériel initial pour les TP2 et suivants sur le Site moodle NF1 ,:

Espace projets –TP/ Source pour les TP 2-5
Ce fichier contient les sources d'un projet java, l'embryon de la grammaire Logo, la librairie antLR, les fichiers batch Windows de génération du code, etc.

DOCUMENTATION ANTLR

Site :	http://www.antlr.org/
Documentation :	ANTLR v4, (Getting Started with ANTLR v4), site ANTLR,
API :	http://www.antlr.org/api/Java/index.html
Livre :	The definitive ANTLR reference, Bibliothèque UTC

DOCUMENTATION LOGO

Manuel Logo : site moodle NF11, Espace projets –TP/Documentations/Manuel LOGO,

DOCUMENTATION JAVA

http://docs.oracle.com/javase/tutorial/

ETAPE 1: INTRODUCTION

CONFIGURATION D'ECLIPSE

- Télécharger le fichier logo-fx.zip à partir du site moodle NF11 :
  Espace projets –TP/Outils TP2/logo-fx.zip

- Créer un projet java : exemple logo-fx. Copier dans le projet au niveau de sa racine les répertoires css, grammar, icon, lib, programs et les deux fichiers batch. Copier dans le répertoire src les répertoires logogui, logomodel et logoparsing.

- Ajouter au projet la librairie antlr-4.9.1-complete.jar. Pour cela sélectionner le fichier lib/antlr-4.9.1-complete.jar et avec le menu contextuel :

  Build Path/Add to Build Path.

- L'ensemble des fichiers doit ressembler à celui de la Figure 1.

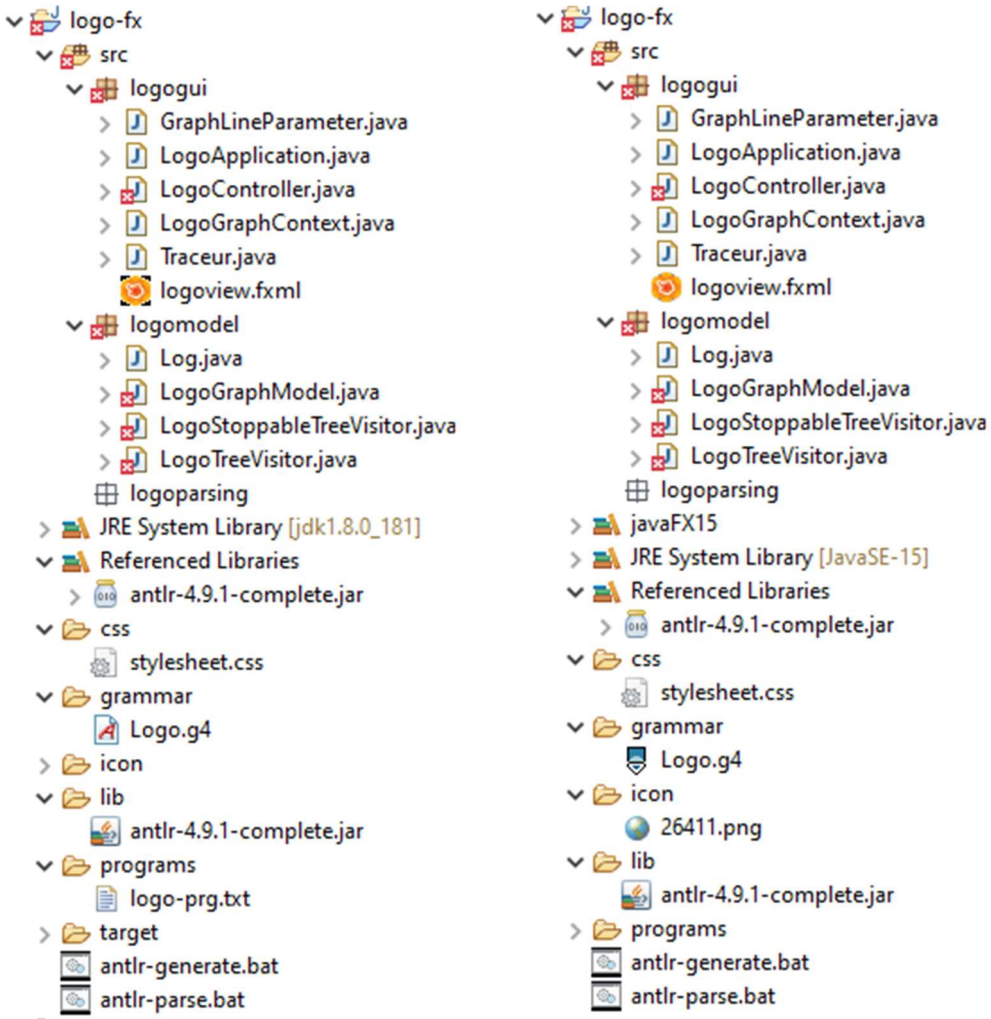


FIGURE 1 : APERÇU DU PROJET ECLIPSE – JAVA 8 – JAVA 15

- Les erreurs de compilation disparaîtront dès la première génération de code qui se fera dans le répertoire logoparsing. Le fichier batch antlr-generate permet de générer les fichiers java à partir de la grammaire grammar/Logo.g4. Le fichier batch antlr-parse

permet de visualiser l'arbre de dérivation du programme programs/logo-prg.txt. Ces deux fichiers doivent être lancés dans une console en dehors de l'environnement Eclipse. Il faut avoir raffraîchi les fichiers du projet pour que l'exécution de antlr-parse fonctionne correctement.

CONFIGURATION D'ECLIPSE - JAVA 15

Dans l'environnement Eclipse, dernière version (4.18) et java 15, il faut suivre les indications d'installation du fichier « Installation de Java FX (pour les tp 2 - 5) ». Notemment :

- installer la user librairy JAVAFX 15 dans Eclipse et la lier au projet.
- Ajouter les VM arguments à la classe programme (contenant le main) : LogoApplication du répertoire logogui.

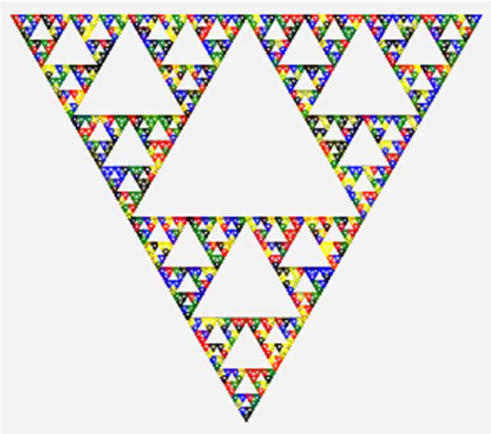
AUTRES CONFIGURATIONS

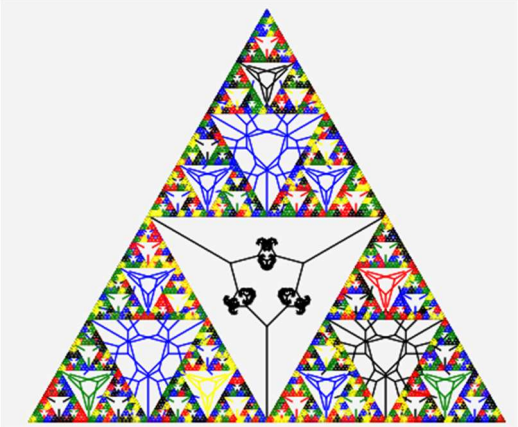
Pour les autres IDE, il faut savoir attacher la user librairy JAVA FX 15 au projet java. Il faut également ajouter les VM arguments au programme LogoApplication.

Pour les OS, Linux, MAC, il faut essayer les équivalents sh des batch Windows. Voir les fichiers de batch-linux. Adapter, si besoin, les répertoires pour que les fichiers générés se retrouvent dans le package logoparsing. Sinon il est possible d'adapter le header du fichier grammaire Logo.g4 et les imports des classes de base qui nécessitent les fichiers générés.

PREMIERS TESTS

- Exécuter la commande antlr-generate dans une console externe. Rafraîchir le source (src) dans Eclipse (File/Refresh ou F5 sous Eclipse) pour voir les classes générées dans le package logoparsing. Exécuter la commande antlr-parse dans une console externe.
- Etudier la grammaire introductive Logo4.g du répertoire grammar.
- Exécuter logogui.LogoApplication : sélectionner ce fichier et à partir du menu contextuel, Run/Java Application sous Eclipse). Taper un programme conforme à la grammaire dans la zone programme. Exécuter. L'application permet de visualiser l'arbre de dérivation et le résultat de l'interprétation du programme Logo.
- Ajouter l'instruction 'tg' expr #tg dans la grammaire Logo.g4 comme alternative dans la règle instruction. Ajouter une ou deux instructions au programme programs/logo-prg.txt. Générer les classes. Parser le programme.
- Exécuter LogoApplication et recopier le programme logo dans le panel de gauche. Exécuter. L'arbre de dérivation doit être conforme. Il n'y a pas encore d'effet graphique.
- Ajouter la méthode visitTg (faire générer la méthode - source/override/implement methods) à la classe LogoTreeVisitor. Ajouter une méthode nécessaire à la classe Traceur. Relancer LogoApplication.
- Etudier le document Manuel Logo pour avoir une idée de ce langage.
- Ajouter les commandes de la tortue à la grammaire : lc (lève crayon), bc (baisse crayon), re (recule), fpos (fixe la position), fcc (fixe la couleur du crayon), fcap (fixe le cap de la tortue). Tester avec l'arbre de dérivation.
- Ajouter les méthodes de visite du LogoTreeVisitor pour chaque règle ajoutée : visitLc, visit… Adapter la classe Traceur pour chaque visite.
- Concevoir un programme permettant de tester toutes les méthodes.
- Dans plusieurs semaines vous pourrez obtenir les résultats suivants :





D'après : Ulysse Brehon et Antoine Courdavault-Duprat (2019)