NF11 – TP2 : GENERATION D'ANALYSEUR LEXICAL ET SYNTAXIQUE (4 SEANCES)

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 une table des symboles 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 le TP2

Site moodle NF11, Espace projets –TP/Outils TP2/tp2.zip
Ce fichier contient l'embryon des grammaires LOGO, la librairie antLR et les fichiers batch de génération du code.

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

DOCUMENTATION JAVA

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

ETAPE 1: INTRODUCTION

CONFIGURATION D'ECLIPSE

- Créer un projet Java dans l'environnement Eclipse (ex : tp2)
- Télécharger le fichier tp2.zip à partir du site NF11 ;
- Décompacter le répertoire src de l'archive dans le répertoire source (tp2/src) d'Eclipse ;
- Créer un répertoire lib sous ce projet (tp2/lib) ; décompacter le répertoire lib de l'archive dans le répertoire lib du projet Eclipse ;
- Installer la librairie ANTLR (antlr-4.0-rc-1-complete.jar) dans le classpath du projet
    - Build Path/add to Build Path
- L'ensemble des fichiers doit ressembler à la Figure 1.

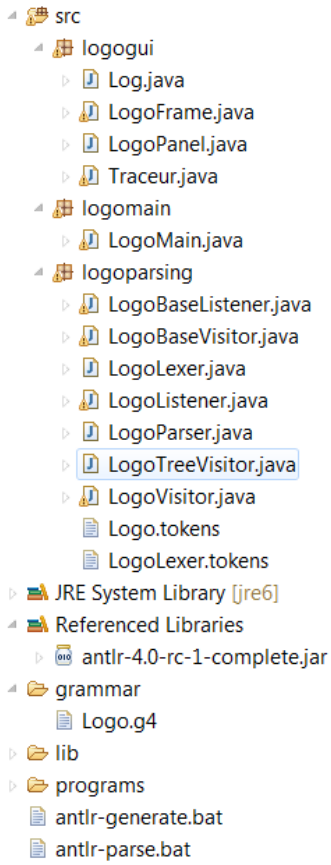

FIGURE 1 : APERÇU DU PROJET ECLIPSE

- Le fichiers batch antlr-generate permet de générer les fichiers java. Le fichier batch antlr-parse permet de visualiser l'arbre de dérivation du programme logo-prg. Ces deux fichiers doivent être lancés dans une console en dehors de Eclipse.

PREMIERS TESTS

- Etudier de la grammaire introductive Logo4.g du répertoire grammar.
- Regarder la méthode testParsing de LogoMain et exécuter LogoMain
- Exécuter antlr-parse dans une console.
- Ajouter l'instruction 'tg' INT #tg dans la grammaire Logo.g4 comme alternative dans la règle instruction. Ajouter une ou deux instructions au programme.
- Générer les classes ; parser le programme
- Commenter testParsing et décommenter la méthode ui de LogoMain.
- Exécuter ui et Recopier le programme logo dans le panel de gauche. Exécuter. L'application permet de visualiser le résultat de l'interprétation du programme Logo.
- Ajouter une méthode visitTg (faire générer la méthode - source/override/implement methods) à la classe LogoTreeVisitor. Ajouter une méthode nécessaire à la classe Traceur. Réexécuter ui.
- Etudier le document Manuel Logo pour avoir une idée de ce langage.

ETAPE 2 : LEXER, PARSER, INTERPRETEUR DU LANGAGE LOGO

De façon cyclique :

ajouter une règle à la grammaire du langage LOGO ou réorganiser les règles afin d'ajouter une nouvelle fonctionnalité.

Faire générer les classes.

Compléter les classes LogoTreeWalker et Traceur. Tester.

TESTER chaque ajout un à un.

Il ne s'agit pas de représenter intégralement la grammaire de Logo, mais d'implémenter les éléments les plus caractéristiques.

REPRESENTER

- Les commandes de la tortue : lc, bc, ve, re, fpos, fcc.
- Les expressions arithmétiques (des exemples se trouvent sur le site AntLR) utilisant les opérateurs *, /, +, -, les parenthèses et la fonction hasard.
- Réorganiser la grammaire pour utiliser les expressions arithmétiques dans les instructions d'un programme. Ex av hasard 100.

A RENDRE SUR MOODLE

La grammaire Logo.g4

Question :

- Comment sont interprétés : av hasard 200 + 100 et av 200 + hasard 100. Pourquoi ?

ETAPE 3 : STRUCTURES DE CONTROLE

Dans les étapes 3 et 4 il faudra créer éventuellement d'autres classes.

REPRESENTER

- les expressions booléennes.
- les identificateurs, l'affectation. Créer une classe pour la table des symboles.
- L'alternative : (si).
- La répétition (répète, tant que). La variable loop.
- Les variables locales à un bloc : leur déclaration utilise le mot-clé LOCALE. Dans le cas contraire une variable est considérée globale au programme.

A RENDRE SUR MOODLE

La grammaire Logo.g4

Le code source de la classe d'une table de symboles

ETAPE 4 : PROCEDURES ET FONCTIONS

REPRESENTER

- Les procédures.
- Les fonctions. Etudier l'appel de procédures ou fonctions. Quel est le problème ?
- Les procédures récursives.

VERIFIER

- L'appel à une procédure non déclarée et l'arité d'une procédure ou d'une fonction.
- Qu'une variable locale ne peut avoir le même nom qu'un paramètre.
- Qu'une variable globale n'est pas utilisée dans une procédure ou une fonction.

EVALUATION

SOUTENANCE ORALE (DERNIERE SEANCE)

Faire une démonstration de l'interpréteur à partir d'un ou plusieurs programmes LOGO. Illustrer les différentes caractéristiques implémentées dans la grammaire.

DOCUMENTS A FOURNIR EN MEME TEMPS QUE LA SOUTENANCE ORALE

Le fichier contenant la grammaire.

Compte-rendu de quelques pages explicitant :

- la modélisation de la table des symboles et le fonctionnement de l'interpréteur (ex : structure des tables, appel et interprétation des procédures, passage de paramètres, etc.)
- la façon dont les procédures ont été traitées (passage des paramètres, appel d'une procédure dans une procédure, procédure récursive, fonction).
- les éléments non traités.
- les vérifications sémantiques.