Ce TP porte sur l'analyse LL(1) du langage INIT qu'on a étendu avec des listes (cf spécification du langage) : l'analyseur à réaliser est un analyseur syntaxique descendant *récursif*. L'analyseur lexical vous est fourni (pour le regénérer, `ant genAnLex` et `ant genAnSynt`).

1 Description du matériel fourni

Copiez chez vous l'archive `tp3.tgz` qui est sur le portail. Elle contient la structure de projet déjà utilisée pour les TP1 et 2.

Inutile de modifier le répertoire `spec` qui contient :
- une spécification d'analyseur lexical pour INIT ;
- un `.cup` qui ne sert à rien d'autre que générer le type énuméré `TypeSymboles`.

Le répertoire `src` contient un paquetage `init` contenant divers fichiers et paquetages que vous connaissez déjà depuis les TP précédents :
- paquetages `executeurs` et `testeurs` contenant les classes principales lancées par les scripts ;
- paquetage `analyseurs` contenant ce qu'il faut pour l'analyse syntaxique de INIT, notamment un squelette de fichier `ParserLL1Init.java`.

Les scripts déjà utilisés dans le TP2, `execEnLigneX.sh`, `execSurFichierX.sh` et `execTestsX.sh`, sont fournis (pour `X` valant `AnalyseurLexical` ou `AnalyseurSyntaxique`).

2 Analyse syntaxique récursive descendante

2.1 Une grammaire LL(1) pour Init étendu

On étend le langage INIT avec un nouveau type de données : les *listes d'entiers*. Suite à cet ajout, le langage INIT n'est plus rationnel, ce qui justifie sa reconnaissance par des mécanismes propres à l'analyse syntaxique.

Une grammaire LL(1)[1] pour les programmes INIT est donnée en annexe A. Son axiome est *programme*. Ses terminaux sont en majuscules, ses non terminaux en minuscules. On donne en annexe B la table d'analyse de cette grammaire, où les productions sont représentées par leur numéro dans la grammaire et *erreur* est représenté par « E ».

2.2 Travail à réaliser

Vous devez compléter la classe `init.analyseurs.ParserLL1Init` pour obtenir un analyseur récursif complet. Telle qu'elle vous est donnée l'analyseur ne reconnaît qu'un en-tête INIT (le vérifier avec `execEnLigneAnalyseurSyntaxique.sh`). Les autres fichiers ne **doivent pas** être modifiés. La compilation se fera par `ant compil`.

Il est *fortement recommandé* d'adopter une mise au point itérative et ascendante ;

1. définir dans l'analyseur syntaxique un non-terminal à la fois, modifier la fonction `parse()`[2] pour tester ce non-terminal, compiler, tester en ligne ;
2. commencer par définir dans l'analyseur syntaxique les non-terminaux qui apparaissent le plus bas possible dans les arbres syntaxiques ;
3. tester au fur et à mesure en alimentant `test/OK` et `test/KO`.

Une fois votre analyseur terminé, vous devez disposer d'une batterie de tests dans `test/OK` et `test/KO` qui s'exécutent avec `execTestsAnalyseurSyntaxique.sh`.

[1] Noter la forme particulière des productions décrivant les listes d'identificateurs.
[2] `analyser()` en cours et en TD.

A Grammaire pour Init étendu

1 *programme* → *entete listeDecl listeInstr*
2 *entete* → *PROG IDENT FININSTR*
3 *listeDecl* → ϵ
4 *listeDecl* → *decl listeDecl*
5 *decl* → *INT listeIdent FININSTR*
6 *decl* → *LIST listeIdent FININSTR*
7 *listeIdent* → *debListeIdent suiteListeIdent*
8 *debListeIdent* → *IDENT*
9 *suiteListeIdent* → ϵ
10 *suiteListeIdent* → *SEPVAR listeIdent*
11 *listeInstr* → ϵ
12 *listeInstr* → *instr listeInstr*
13 *instr* → *affect*
14 *instr* → *lect*
15 *affect* → *IDENT AFF exprAffect FININSTR*
16 *lect* → *READ IDENT FININSTR*
17 *exprAffect* → *ENTIER*
18 *exprAffect* → *liste*
19 *liste* → *DEBLISTE listeElt FINLISTE*
20 *listeElt* → ϵ
21 *listeElt* → *elt listeElt*
22 *elt* → *ENTIER*
23 *elt* → *liste*

B Table d'analyse

	PROG	*INT*	*LIST*	*READ*	*IDENT*	*ENTIER*	*AFF*	*FININSTR*	*SEPVAR*	*DEBLISTE*	*FINLISTE*	*EOF*
programme	1	E	E	E	E	E	E	E	E	E	E	E
entete	2	E	E	E	E	E	E	E	E	E	E	E
listeDecl	E	4	4	3	3	E	E	E	E	E	E	3
decl	E	5	6	E	E	E	E	E	E	E	E	E
listeIdent	E	E	E	E	7	E	E	E	E	E	E	E
debListeIdent	E	E	E	E	8	E	E	E	E	E	E	E
suiteListeIdent	E	E	E	E	E	E	E	9	10	E	E	E
listeInstr	E	E	E	12	12	E	E	E	E	E	E	11
instr	E	E	E	14	13	E	E	E	E	E	E	E
affect	E	E	E	E	15	E	E	E	E	E	E	E
lect	E	E	E	16	E	E	E	E	E	E	E	E
exprAffect	E	E	E	E	E	17	E	E	E	18	E	E
liste	E	E	E	E	E	E	E	E	E	19	E	E
listeElt	E	E	E	E	E	21	E	E	E	21	20	E
elt	E	E	E	E	E	22	E	E	E	23	E	E