h e p i a
Haute école du paysage, d'ingénierie et d'architecture de Genève

hepia
Ingénierie des Technologies de l'information
Technique de compilation - TCP

Labo 2 : Un analyseur SGML

MINELLI Michaël

```
#############################################################
#          Labo2 : JFlex et CUP -- Analyseur SGML           #
#############################################################
# Auteur : Michaël MINELLI                                  #
# Classe : ITI Jour 2ème                                    #
#                                                           #
# Fichier testé : test.txt                                  #
#############################################################
#                          Résultat                         #
#############################################################
+A C +D -D -A
#############################################################
```

Table des matières

1 Réalisation

La réalisation s'est déroulée en deux grandes phases. Il s'agit tout simplement des phases JFlex et cup.

1.1 Phase JFlex

Lors de la phase JFlex il a été question de déterminer comment fractionner la détection afin de n'omettre aucune détection. La première chose qui fut visible, est que les balise de fermeture serait toujours de la même forme ("</X>") sans possibilité d'erreur selon ce qui a été dit lors de la présentation du Labo.

Après, il a fallu analyser les balises ouvrantes et auto-fermantes, car elles doivent pouvoir contenir des identificateurs tout en faisant attention qu'elles ne comportent pas d'erreur (guillemet non fermé ou simplement texte ne devant pas être là). Ces balisent ont donc été fractionné en trois morceaux. Le premier est l'ouverture de balise ("<X") et le dernier est la fermeture (">" ou "/>"). La deuxième partie, quant à elle, est celle pouvant contenir les identificateurs ou les erreurs. La détection d'un identificateur ('a="texte"') a donc été ajoutée.
A ce moment le dernier problème restant était donc la détection d'erreur. Celle-ci s'est faite part l'intermédiaire de la détection de tous les caractère qui n'on pas été détecté par une autre règle. Cela voulait dire deux choses. La première est que ce n'était pas forcément une erreur, mais cela pouvait aussi s'avérer être le texte entre les balises il faudrait donc déterminer par la suite (dans le cup) si cela est vraiment une erreur. La deuxième qu'il fallait ignorer la détection des caractères non pris en erreurs (espace, retour chariot, etc.) ainsi que les identificateurs.

La partie JFlex a été testée tout d'abord sans le cup avec de simple affichage println.

Voici le code final du JFlex :

1.1.1 Fichier JFlex

```
1  /* Michaël Minelli
2   * Technique de compilation
3   * ITI Jour 2ème
4   * Labo 2 : Analyseur SGML
5   */
6  import java_cup.runtime.*;
7  import java.util.Vector;
8  %%
9  %class Labo2
10 %line
11 %column
12 %cup
13 %{
14 %}
15
16 LETTRE=[A-Z]
17 LETTRE_MIN=[a-z]
18 FERMANTE=\<\/{LETTRE}\>
19 DEBUT=\<{LETTRE}
20 ID={LETTRE_MIN}\=\".*\"
21 FIN_AUTOFERMANTE=\/\>
22 FIN=\>
23 IGNORE=(\ |\t|\n|{ID})
24
25 %%
26
27 {FERMANTE} {return new Symbol(sym.FERMANTE, String.valueOf(yytext().toCharArray()[2]));}
28 {DEBUT}   {return new Symbol(sym.DEBUT, String.valueOf(yytext().toCharArray()[1]));}
29 {FIN_AUTOFERMANTE} {return new Symbol(sym.FIN_AUTOFERMANTE);}
30 {FIN} {return new Symbol(sym.FIN);}
31 {IGNORE} {;}
32 . {return new Symbol(sym.TEXT_OR_ERROR);}
```

Listing 1 – `labo2.flex`

1.2 Phase Java Cup

Une première solution fut envisagé puis écarté, car ne correspondait pas au cahier des charges. Cette solution mettait en place des variables bouléaines et une pile afin de tester les fermetures de balises. Celle-ci était tout à fait réalisable, mais ne nécessitait pas de cup simplement du JFlex au contraire du but de ce labo.

Dans la solution retenue il a fallu premièrement concaténé les erreurs potentiel, car dans le JFlex tout caractère était pris comme tel donc il était possible d'avoir plusieurs (sans savoir exactement combien à l'avance) erreur potentiels à la suite.
Tout était presque mis en place il ne restait plus qu'à détecter les balises ouvrantes et autofermantes en entier, car elles ont été fractionnée par le JFlex. Pour le besoin de la détection d'erreur pour chacune des balises, deux variables non-terminal ont été crée. La première si aucune erreur n'est présente et que l'arrêt de l'affichage n'est pas à vrai, la balise sera affichée et pour les balises ouvrante la valeur de celle-ci mise en mémoire. La deuxième s'il y a une erreur, cela affiche la lettre avec la mention "(erreur)" et met la variable indiquant la fin de l'affichage à vrai.
Enfin, la grammaire rassemble le tout en vérifiant que la lettre des balises ouvrantes correspond à la lettre des balises fermantes.

Voici le code final du cup :

1.2.1 Fichier cup

```
/* Michaël Minelli
* Technique de compilation
* ITI Jour 2ème
* Labo 2 : Analyseur SGML
*/
import java_cup.runtime.*;
import java.util.*;

action code {:
    //Stop l'affichage si une erreur a été trouvée
   public Boolean stopExec = false;

    //Test que le balise se soit fermée avec le meme type de balise
   public void testBaliseFermante(String l, String m) {
      if (!stopExec) {
         System.out.printf("-" + m + " ");
         if (!l.equals(m)) {
            System.out.printf("(erreur) ");
            stopExec = true;
         }
      }
   }
:};

terminal String DEBUT, FERMANTE;
terminal FIN_AUTOFERMANTE, FIN, TEXT_OR_ERROR;

non terminal String baliseAutoFermante, baliseAutoFermanteWithError, balise, baliseWithError;
non terminal textOrError, sgml, sgml_withoutError;

//Tout le code
sgml ::= balise:l sgml FERMANTE:m sgml {: testBaliseFermante(l, m); :}
         | baliseWithError sgml FERMANTE sgml
         | baliseAutoFermante sgml
         | baliseAutoFermanteWithError sgml
         | textOrError sgml_withoutError
         | ;

//Permet de contourné une érreur avec les textOrError qui fesait de la grammaire une grammaire ambigue
sgml_withoutError ::= balise:l sgml FERMANTE:m sgml {: testBaliseFermante(l, m); :}
         | baliseWithError sgml FERMANTE sgml
         | baliseAutoFermante sgml
         | baliseAutoFermanteWithError sgml
         | ;

//Balises auto-fermante
baliseAutoFermante ::= DEBUT:n FIN_AUTOFERMANTE {: if (!stopExec) { System.out.printf(n + " "); RESULT = n; } :};
baliseAutoFermanteWithError ::= DEBUT:n textOrError FIN_AUTOFERMANTE {: if (!stopExec) { System.out.printf(n + " ↩
     (erreur)"); stopExec = true; } :};

//Balises ouvrantes
balise ::= DEBUT:n FIN {: if (!stopExec) System.out.printf("+" + n + " "); RESULT = n; :};
baliseWithError ::= DEBUT:n textOrError FIN {: if (!stopExec) { System.out.printf("+" + n + " (erreur)"); stopExec = ↩
     true; } :};

//Fais en sort de réduite plusieurs TEXT_OR_ERROR à  la suite par un seul textOrError
textOrError ::= textOrError TEXT_OR_ERROR | TEXT_OR_ERROR;
```

Listing 2 – `labo2.cup`

2 Compilation

Le makefile a été concu de sorte qu'il n'y ai rien d'autre a faire qu'un "make". Voici le contenu du fichier "LISEZMOI" :

1. Ouvrer la console dans le dossier contenant les fichier du labo
2. Effectuez un « make » avec l'option non-obligatoire « TESTFILE=... » (ou les ... est le nom du fichier à analyser). Si cette option n'est pas indiquée cela prendra comme nom de fichier par défaut : test.txt

Exemples :
a. make
b. make TESTFILE=test_faux.txt

2.1 Code du makefile

```
1  JAVA=java
2  JAVAC=javac
3  JFLEX=jflex
4  JAVACUP=java-cup-11a.jar
5  CLASSPATH=$(JAVACUP):.
6
7  FILE_FLEX=Labo2.flex
8  FILE_CUP=Labo2.cup
9  FILE_JAVA_NAME=Labo2
10 FILE_TEST_PRG_NAME=test
11
12 TEST_CLASS=test
13
14 FILE=test.txt
15 ifdef TESTFILE
16    FILE=$(TESTFILE)
17 endif
18
19 all : $(FILE) sym.class parser.class $(FILE_JAVA_NAME).class $(FILE_TEST_PRG_NAME).class
20    $(JAVA) -classpath $(CLASSPATH) $(TEST_CLASS) $(FILE)
21
22 $(FILE_JAVA_NAME).java : $(FILE_FLEX)
23    $(JFLEX) $(FILE_FLEX)
24
25 sym.java parser.java : $(FILE_CUP)
26    $(JAVA) -jar $(JAVACUP) $(FILE_CUP)
27
28 %.class : %.java
29    $(JAVAC) -classpath $(CLASSPATH) $<
30
31 clean :
32    rm -rf *class *~ parser.java sym.java $(FILE_JAVA_NAME).java
```

Listing 3 – `makefile`