LINGI2132 Assignement 2 Report

Julien De Coster Xavier Crochet

March 15, 2013

1 Multi Line Comments

1.1 Handwritten scanner

Add, in the '/' case, that if another '*' is scanned, we have to skip all the characters until we scan '*/'.

1.2 JavaCC generated scanner

Add in the j–.jj file a grammar rule that switch in the state *multiline* if '/*' is encountered. We move from this state only if '/*' is scanned.

2 Conditional Expression

2.1 Implementation

We add the *QM '?'* token and the *COLON ':'* one, create a class JConditionalExpression and only implements the constructor and the writeToStdOut method as asked in the *Ènoncé*. To do so, we create the method *conditionalExpression*. This method invokes the lower level, *conditionalAndExpression* and if it scans a QM token, create an *assignmentExpression*, check whether there is a COLON token, *recursively* call itself and return a new JConditionalStatement with the previous created parameters.
A JConditionalStatement is a Statement with the following parameters:

- JExpression *condition* : the condition to test
- JExpression *then part* : the expression to branch to if *condition* is true
- JExpression *else part* : the expression to branch to if *condition* is false (this expression can be another JConditionalExpression)

2.2 Difference between the handwritten scanner an the other one

2.3 Tests

We test tricky case as follow :

- Comparison in the condition

- Boolean in the condition
- Conditional expression with conditional expression in the else part
- ...

3 For Statement

3.1 Implementation

Here we have to support two type of for loop ;

1. The basic one : for ([forInit] ; [expression] ; [forUpdate]) statement
2. The enhanced one : for (FormalParameter : Expression) Statement

For the first one, we have to create a forInit and a forUpdate methode to parse the concerned expression. Concerning forInit, we have to handle the *final* modifier and multiples variable declarations. Concerning forUpdate, we just have to handle multiple statements. Notice that we have to parse the *final* modifier in the variableDeclaration function and that modifiers are already supported by the class. However, we have to add *final* modifier support into the JFormalParameter class in order to parse it when encountered.

3.2 Difference between the handwritten scanner an the other one

In j–.jj we have to use *LOOKAHEAD(PATTERN)* in order distinguish the two for loops.

3.3 Tests

- Final modifier in *forInit/formalParameter*
- Multiple variable declaration in *forInit*
- Multiple statement in *ForUpdate*
- Empty *basicForLoop* (for(;;;))

4 Conclusion