ECE 664: Homework 8

Rahul Deshmukh

EBNF.jjt:

```
/**
 EBNF grammar file for ECE 664 HW8
 Useful for jjtree: https://mathcs.clarku.edu/~fgreen/courses/cs230/examples/JJTreeExamples/
*/

options{
    MULTI = true;
 STATIC = false;
 NODE_DEFAULT_VOID=true;
}

PARSER_BEGIN(EBNF)
 import java.io.*;
 class EBNF {
   public static void main(String[] args)
   throws ParseException,TokenMgrError {
     EBNF parser = new EBNF(System.in);
     /*Parse the input*/
  try {
     ASTSTART root = parser.Start();
     System.out.println("** Text parsed successfully **");
     /* Print the jjtree */
  root.dump("");
  } catch(Exception e ) {
  System.out.println("!! Text could not be parsed !!");
  System.out.println(e.getMessage());
  e.printStackTrace();
  }
   }
 }
PARSER_END(EBNF)

/** Lexical Specs */
SKIP: {" "}
SKIP: { "\n"|"\r\n"|"\r"}
TOKEN : {< DETERMINER:"the" | "every" | "this" | "a" | "an" >}
TOKEN : {< NOUN:"time" | "arrow" | "flies" | "widget" | "cat" >}
TOKEN : {< PRONOUN:"i" | "we" | "this" | "you" | "us" >}
TOKEN : {< VERB:"is" | "are" | "have" | "has" | "make" | "time" | "flies" |
    "build" | "stores" | "buying" | "like" > }
TOKEN : {< ADJ: "time" | "big" | "high quality" | "fast" | "large" | "precise" |
  "high" >}
TOKEN : {< PREPOSITION: "with" | "without" | "for" | "about" | "at" | "after" |
      "between" | "but" | "below" | "in" | "into" | "like" |
      "except" | "following" | "on" | "than" | "over" | "near"|
      "of" | "like" | "near" | "since" | "up" | "upon" | "within" |
      "by" >}
TOKEN : /* words */ {  < WORD: (<LETTER>)+ > | < #LETTER: ["a"-"z","A"-"Z"] > }
TOKEN : /* Separators */ {< PERIOD : "." >}

/** Parser Specs */
ASTSTART Start() #START:{}
{  ( Sentence() <PERIOD> )+< EOF >   {return jjtThis;}}
```

```
54
55 void Sentence() #SENTENCE:{}
56 {  ( NP())? VP() }
57
58 void NP() #NP:{}
59 {  ( pronoun()  | ((determiner() )? (adj())* CN()) ) (PP())? }
60
61 void CN() #CN:{}
62 {  noun() (noun())?  }
63
64 void PP() #PP:{}
65 {  preposition()  NP() }
66
67 void VP() #VP:{}
68 {  verb() NP() }
69
70 // LEAF NODES
71 void determiner() #DETERMINER:{ Token t;}
72 {  t = < DETERMINER >  {jjtThis.setLexem(t.image); }}
73
74 void noun() #NOUN:{ Token t;}
75 {  t = < NOUN >  {jjtThis.setLexem(t.image); }}
76
77 void pronoun() #PRONOUN:{ Token t;}
78 {  t = < PRONOUN >  {jjtThis.setLexem(t.image); }}
79
80 void verb() #VERB:{ Token t;}
81 {  t = < VERB >  {jjtThis.setLexem(t.image); }}
82
83 void adj() #ADJ:{ Token t;}
84 {  t = < ADJ >  {jjtThis.setLexem(t.image); }}
85
86 void preposition() #PREPOSITION:{ Token t;}
87 {  t = < PREPOSITION >  {jjtThis.setLexem(t.image); }}
```

ASTADJ.java:

```
1 /* Generated By:JJTree: Do not edit this line. ASTADJ.java Version 4.3 */
2 /* JavaCCOptions:MULTI=true,NODE_USES_PARSER=false,VISITOR=false,TRACK_TOKENS=false,
     NODE_PREFIX=AST,NODE_EXTENDS=,NODE_FACTORY=,SUPPORT_CLASS_VISIBILITY_PUBLIC=true
     */
3 public
4 class ASTADJ extends SimpleNode {
5   public ASTADJ(int id) {
6     super(id);  }
7   public ASTADJ(EBNF p, int id) {
8     super(p, id);  }
9   //My added members
10  String lexem = "";
11  public void setLexem( String lex) {lexem = lex;}
12  public String getLexem() {return lexem;}
13  //Override super class handling
14  public String toString() {return super.toString()+": "+lexem;}}
15 /* JavaCC - OriginalChecksum=84ec3d15f2ae2e00998b624f5d067856 (do not edit this line
     ) */
```

Input text file:

```
time flies like an arrow.
```

Output:

```
** Text parsed successfully **
START
 SENTENCE
  NP
   CN
    NOUN: time
    NOUN: flies
  VP
   VERB: like
   NP
    DETERMINER: an
    CN
     NOUN: arrow
```