Jacob Longar

10-3-2017

Csci 465

Source File

```
import java.io.*;
import java.util.Scanner;
import java.util.regex.Matcher;
import java.util.regex.Pattern;

public class languageScanner
{
  public static void main(String args[]) throws IOException
  {
    try
    {
      //prompting the user
      System.out.print("Please enter the name of the input file: ");
      Scanner input = new Scanner(System.in);
      String inputVal = input.nextLine();

      //file IO
      FileReader fileReader = new FileReader(inputVal);
      BufferedReader bufferedReader = new BufferedReader(fileReader);
      StringBuffer stringBuffer = new StringBuffer();

      String line;
      while ((line = bufferedReader.readLine()) != null) {
        stringBuffer.append(line);
        stringBuffer.append("\n");
      }
      fileReader.close();
      String fileStuff = stringBuffer.toString();
```

```
        //System.out.println(fileStuff);


        //regex variables
        final String regex =
"(\\()|(\\))|(\\[)|(\\])|(\\.)|(\\+)|(\\-)|(\\*)|(<=)|(>=)|(=)|(<>)|(<)|(>)|(:=)|(:)|(;)|(,)|(and)|(array)|(begin)|(do)|(char)|(ch
r)|(div)|(else)|(end)|(if)|(integer)|(mod)|(not)|(of)|(or)|(ord)|(procedure)|(program)|(read)|(readln)|(then)|(var)
|(while)|(write)|(writeln)|(function)|(\\{.*\\})|('.*')|(\".*\")|([a-zA-Z]\\w+)|([a-zA-Z])|([0-9]+\\.[0-9]+)|([0-9]+)";
        final Pattern pattern = Pattern.compile(regex, Pattern.CASE_INSENSITIVE);
        final Matcher matcher = pattern.matcher(fileStuff);
        //matcher is now ready for regex regognition in the while loop below the output file declaration


        //output file declaration
        File fout = new File("a.out");
        FileOutputStream fileOutputStream = new FileOutputStream(fout);


        BufferedWriter bufferedWriter = new BufferedWriter(new OutputStreamWriter(fileOutputStream));


        //initializing line counter
        int count = 0;


        while (matcher.find())
        {
          for (int i = 1; i <= matcher.groupCount(); i++)
          {
            if (matcher.group(i) != null)
            {                   //System.out.println("Group " + i + ": " + matcher.group(i));
              if (i == 53)
                //for finding which line the error is on.
                count++;
              if (i == 1)
                bufferedWriter.write(String.format("%-20s%S\n" , "LPAREN", matcher.group()));
//matcher.group(1)
              if (i == 2)
                bufferedWriter.write(String.format("%-20s%S\n" , "RPAREN", matcher.group()));
              if (i == 3)
```

```
  bufferedWriter.write(String.format("%-20s%S\n" , "LBRACKET", matcher.group()));
if (i == 4)
  bufferedWriter.write(String.format("%-20s%S\n" , "RBRACKET", matcher.group()));
if (i == 5)
  bufferedWriter.write(String.format("%-20s%S\n" , "PERIOD", matcher.group()));
if (i == 6)
  bufferedWriter.write(String.format("%-20s%S\n" , "PLUS", matcher.group()));
if (i == 7)
  bufferedWriter.write(String.format("%-20s%S\n" , "MINUS", matcher.group()));
if (i == 8)
  bufferedWriter.write(String.format("%-20s%S\n" , "TIMES", matcher.group()));
// if (i == 9)
//    System.out.println("<lparen>, " + matcher.group()); //doesn't require a token to be created.
if (i == 9)
  bufferedWriter.write(String.format("%-20s%S\n" , "LESSEQUAL", matcher.group()));
if (i == 10)
  bufferedWriter.write(String.format("%-20s%S\n" , "GREATEREQUAL", matcher.group()));
if (i == 11)
  bufferedWriter.write(String.format("%-20s%S\n" , "EQUAL", matcher.group()));
if (i == 12)
  bufferedWriter.write(String.format("%-20s%S\n" , "NOTEQUAL", matcher.group()));
if (i == 13)
  bufferedWriter.write(String.format("%-20s%S\n" , "LESSTHAN", matcher.group()));
if (i == 14)
  bufferedWriter.write(String.format("%-20s%S\n" , "GREATERTHAN", matcher.group()));
if (i == 15)
  bufferedWriter.write(String.format("%-20s%S\n" , "ASSIGNMENT", matcher.group()));
if (i == 16)
  bufferedWriter.write(String.format("%-20s%S\n" , "COLON", matcher.group()));
if (i == 17)
  bufferedWriter.write(String.format("%-20s%S\n" , "SEMICOLON", matcher.group()));
if (i == 18)
  bufferedWriter.write(String.format("%-20s%S\n" , "COMMA", matcher.group()));
```

```
if (i == 19)
  bufferedWriter.write(String.format("%-20s%S\n" , "AND", matcher.group()));
if (i == 20)
  bufferedWriter.write(String.format("%-20s%S\n" , "ARRAY", matcher.group()));
if (i == 21)
  bufferedWriter.write(String.format("%-20s%S\n" , "BEGIN", matcher.group()));
if (i == 22)
  bufferedWriter.write(String.format("%-20s%S\n" , "DO", matcher.group()));
if (i == 23)
  bufferedWriter.write(String.format("%-20s%S\n" , "CHAR", matcher.group()));
if (i == 24)
  bufferedWriter.write(String.format("%-20s%S\n" , "CHR", matcher.group()));
if (i == 25)
  bufferedWriter.write(String.format("%-20s%S\n" , "DIVIDE", matcher.group()));
if (i == 26)
  bufferedWriter.write(String.format("%-20s%S\n" , "ELSE", matcher.group()));
if (i == 27)
  bufferedWriter.write(String.format("%-20s%S\n" , "END", matcher.group()));
if (i == 28)
  bufferedWriter.write(String.format("%-20s%S\n" , "IF", matcher.group()));
if (i == 29)
  bufferedWriter.write(String.format("%-20s%S\n" , "INTEGER", matcher.group()));
if (i == 30)
  bufferedWriter.write(String.format("%-20s%S\n" , "MOD", matcher.group()));
if (i == 31)
  bufferedWriter.write(String.format("%-20s%S\n" , "NOT", matcher.group()));
if (i == 32)
  bufferedWriter.write(String.format("%-20s%S\n" , "OF", matcher.group()));
if (i == 33)
  bufferedWriter.write(String.format("%-20s%S\n" , "OR", matcher.group()));
if (i == 34)
  bufferedWriter.write(String.format("%-20s%S\n" , "ORD", matcher.group()));
if (i == 35)
```

```
            bufferedWriter.write(String.format("%-20s%S\n" , "PROCEDURE", matcher.group()));
          if (i == 36)
            bufferedWriter.write(String.format("%-20s%S\n" , "PROGRAM", matcher.group()));
          if (i == 37)
            bufferedWriter.write(String.format("%-20s%S\n" , "READ", matcher.group()));
          if (i == 38)
            bufferedWriter.write(String.format("%-20s%S\n" , "READLN", matcher.group()));
          if (i == 39)
            bufferedWriter.write(String.format("%-20s%S\n" , "THEN", matcher.group()));
          if (i == 40)
            bufferedWriter.write(String.format("%-20s%S\n" , "VAR", matcher.group()));
          if (i == 41)
            bufferedWriter.write(String.format("%-20s%S\n" , "WHILE", matcher.group()));
          if (i == 42)
            bufferedWriter.write(String.format("%-20s%S\n" , "WRITE", matcher.group()));
          if (i == 43)
            bufferedWriter.write(String.format("%-20s%S\n" , "WRITELN", matcher.group()));
          if (i == 44)
            bufferedWriter.write(String.format("%-20s%S\n" , "FUNCTION", matcher.group()));
          //(I == 45) tells us that this line is a comment so we don't include it in the stream
          if (i == 46 || i == 47)
            bufferedWriter.write(String.format("%-20s%S\n" , "STRING", matcher.group()));
          if (i == 48 || i == 49)
            bufferedWriter.write(String.format("%-20s%S\n" , "ID", matcher.group()));
          if (i == 50)
            bufferedWriter.write(String.format("%-20s%S\n" , "FLOATNUMBER", matcher.group()));
          if (i == 51)
            bufferedWriter.write(String.format("%-20s%S\n" , "NUMBER", matcher.group()));
        }
      }
    }
    bufferedWriter.close();
}
```

```
  catch (IOException e)
  {
    System.out.println("\nInput file does not exist.");
    System.out.println("Please enter an existing file name and run the program again.");
  }
  getsym();
}

public static void getsym()
{
  try
  {
    FileReader fin = new FileReader("a.out");
    Scanner src = new Scanner(fin);
    while(src.hasNext())
    {
     // System.out.println(src.next());
      if (src.next() == "END")
      {
        if(src.next() == "END")
        {
          if (src.next() != "SEMICOLON")
          {
            System.out.println("Semicolon expected after this function on line: ", count);
            src.previous();
            src.previous();
          }
        }
      }
      if (src.next() == 'LPAREN')
      {
        if (src.find('RPAREN') == false)
        {
```

```
                System.out.println("Cannot find right parenthese on line: ", count);
            }
          }
        }
      }
      catch(IOException e)
      {
        System.out.println("Lister file was not found.");
      }
    }
}
```

Jacob Longar

Output File (correct input)

```
PROGRAM             PROGRAM
ID                  EXAMPLE
LPAREN              (
ID                  INPUT
COMMA               ,
ID                  OUTPUT
RPAREN              )
SEMICOLON           ;
VAR                 VAR
ID                  X
COMMA               ,
ID                  Y
COLON               :
INTEGER             INTEGER
SEMICOLON           ;
FUNCTION            FUNCTION
ID                  GCD
LPAREN              (
ID                  A
COMMA               ,
ID                  B
COLON               :
INTEGER             INTEGER
RPAREN              )
COLON               :
INTEGER             INTEGER
SEMICOLON           ;
BEGIN               BEGIN
IF                  IF
ID                  B
EQUAL               =
NUMBER              0
THEN                THEN
ID                  GCD
ASSIGNMENT          :=
ID                  A
ELSE                ELSE
ID                  GCD
ASSIGNMENT          :=
LPAREN              (
ID                  B
COMMA               ,
ID                  A
```

```
MOD                 MOD
ID                  B
RPAREN              )
END                 END
SEMICOLON           ;
BEGIN               BEGIN
READ                READ
LPAREN              (
ID                  X
COMMA               ,
ID                  Y
RPAREN              )
SEMICOLON           ;
WRITE               WRITE
LPAREN              (
ID                  GCD
LPAREN              (
ID                  X
COMMA               ,
ID                  Y
RPAREN              )
RPAREN              )
END                 END
PERIOD              .

.
```

Jacob Longar

Correct Input file:

```
program example(input,output);
var x,y:integer;
function gcd(a,b:integer):integer;
begin{gcd}
if b=0then gcd:=a else gcd:=(b,a mod b)
end; {gcd}
begin{example}
read(x,y);
write(gcd(x,y))
end;
```

Incorrect Input file:

```
program example(input,output);
var x,y:integer;
function gcd(a,b:integer:integer;
begin{gcd}
if b=0then gcd:=a else gcd:=(b,a mod b
end; {gcd}
begin{example}
read(x,y);
write(gcd(x,y))
end.
```

Incorrect Input file command line outputs:

Semicolon expected after this function on line: 10

Cannot find right parenthese on line: 2

Cannot find right parenthese on line: 4