Assign-Parser-DESCR-Fa23

Version: Refer to the version in the file name

Contents

1 Description of the Parser and Respective Grammar

1.1 Description

The Java program "**Parse.java**", which is given to you on D2L, is based on the one given on page 175 of the textbook and on what was covered in class. The parser in the textbook is implementing only the rules 5-7 of the following grammar. The parser, Parse.java, however, implements all the rules, except the ones highlighted in blue.

Modify the Parse.java program to incorporate the following changes.

a. In this parser, use your own lexical analyzer, which you have completed, instead of the one which is already used in this parser program. This will be 10% of the points.
b. If you wish not to use your own lexical analyzer, just modify the given parser to address the requirements of the program:(90%)
c. Complete the implementation of rule 9.
d. Make the program **more readable** by adding, deleting, or updating the comments, identifier names and modules, as appropriate. Also, make sure that consistent line spacing and indentations are used throughout the program.

1.2 Grammar

```
1) <program> → PROGRAM <statement>{;<statement>} END
2) <statement> → <output> | <assign> | <input> | <selection>
3) <output> → print (<expr>)
4) <assign> → IDENT = <expr>
5) <expr> → <term> {(+ | -) <term>}
6) <term> → <factor> {(* | /) <factor>}
7) <factor> → IDENT | INT_LIT | ( <expr> )
8) <input> → read (IDENT)
9) <selection> → if (<expr>) then <statement> [else <statement>]
```

<program> is the starting symbol of the grammar.

2 Tokens

ADD_OP
ASSIGN_OP
DIV_OP
ELSE_KEYWORD
END_KEYWORD
END_OF_FILE
IDENT
IF_KEYWORD
INT_LIT
LEFT_PAREN
MULT_OP
PRINT_KEYWORD
PROGRAM_KEYWORD
READ_KEYWORD
RIGHT_PAREN
SEMICOLON
SUB_OP
THEN_KEYWORD

3 Parser Input and Output Files

Your lexical analyzer must read the source program statements from a file called **sourceProgram.txt**, and to generate the appropriate tokens for the parser.

The parser output must be stored on the **parseOut.txt** file.

4 Example Source Program and Corresponding Parsed Result

4.1 Source program

```
PROGRAM
    sumTotal = (sum / 47 ) + total
    Total = (sum+ 407 ) / 123
END
```

4.2 The result of parsing the above program

John D. Student, CSCI4200, Fall 2023, Parser
****

PROGRAM
  sumTotal = (sum / 47) + total
  Total = (sum+ 407) / 123
END
****

Next token is: PROGRAM_KEYWORD
Enter <program>
Next token is: IDENT
Enter <statement>
Enter <assign>
Next token is: ASSIGN_OP
Next token is: LEFT_PAREN
Enter <expr>
Enter <term>
Enter <factor>
Next token is: IDENT
Enter <expr>
Enter <term>
Enter <factor>
Next token is: DIV_OP
Exit <factor>
Next token is: INT_LIT
Enter <factor>
Next token is: RIGHT_PAREN
Exit <factor>
Exit <term>
Exit <expr>
Next token is: ADD_OP
Exit <factor>
Exit <term>
Next token is: IDENT
Enter <term>
Enter <factor>

Exit <factor>
Exit <term>
Exit <expr>
Exit <assign>
Exit <statement>

Error - missing semicolon
Next token is: IDENT
Enter <statement>
Enter <assign>
Next token is: ASSIGN_OP
Next token is: LEFT_PAREN
Enter <expr>
Enter <term>
Enter <factor>
Next token is: IDENT
Enter <expr>
Enter <term>
Enter <factor>
Next token is: ADD_OP
Exit <factor>
Exit <term>
Next token is: INT_LIT
Enter <term>
Enter <factor>
Next token is: RIGHT_PAREN
Exit <factor>
Exit <term>
Exit <expr>
Next token is: DIV_OP
Exit <factor>
Next token is: INT_LIT
Enter <factor>
Exit <factor>
Exit <term>
Exit <expr>
Exit <assign>
Exit <statement>

Next token is: END_KEYWORD
Exit <program>
Parsing of the program is complete!
****

5 The Test Program to Parse and Submit

Parse the following program, keeping the spacing and format of the program intact. I will test your program using the following and other programs.

```
PROGRAM
    read(count);
    total = 205;
        Area = (length+width) * 20
    C    =        age  - 15 * (D / 8);
    print(C+Area);
    if (C) then C = 99;
    if (count) then average = total/count else print(count)
END
```

6 How to Structure and Organize Your Program

a. In your IDE (e.g., Eclipse) create a Java project called **CSCI4200Fa23Parse**.

b. Store your program, **Parse**, in the **default** package of **src** within the **CSCI4200Fa23Parse** project.

c. Store the source program to be parsed in an input file called **sourceProgram.txt**.

d. Store the sourceProgram.txt in the same project, **CSCI4200Fa23Parse**. This will allow you to open the input file as “sourceProgram.txt” without specifying a path.

e. Store your output in a text file called **parseOut.txt**, which must be created in the same project. You must create and open the output file as “parseOut.txt” without specifying a path.

f. On the output, display your name, class information and a program title, followed by a line of 80 asterisks (*).

g. On the output, display the source program being parsed followed by the parser result, as shown in the example above.

h. On the output, display a blank line after parsing each line of the source program, as shown in the above example.

i. The output of your parser ought to be similar to the EXAMPLE output above. Please note that the above source program and the corresponding parser output are only examples. The actual output may be different based on the source program being parsed.

j. Include the following, as comments, at the top of your program:

    a. Your Complete Name:

    b. Class: CSCI 4200, Fall 2023

    c. Program: Parser

d. Did you use your own lexical analyzer? (Yes or No):

If Yes, submit your original lexical analyzer too.

7 What to Turn in

a. Parse.java

b. sourceProgram.txt

c. parseOut.txt

d. Lex.java, if you used your own lexical analyzer

8 Grading Rubrics

Category	Description	Points
1. Submissions	Files are submitted as described	5
2. Program Readability	Sufficient and readable comments (don't overdo it!) Readable program – descriptive identifier names; use of constant identifiers; appropriate program structures such as loops, decisions, etc.; appropriate indentations and spacing, etc. All the project-, class-, input file- and output file names must be exactly as specified. Your name and class information must be stated as comments at the top of the parser code.	15
3. Program logic and output	Use of correct straightforward (simple) and understandable logic Correct and complete results Correct output format	70
4. Input file	Correct contents Correct format (as specified by the requirements)	5
5. Output file	Contents	5