

San José State University
Department of Computer Science

CS 153

Concepts of Compiler Design

Section 1
Fall 2020
Instructor: Ron Mak

Assignment #2

Assigned: Thursday, August 27
Due: Thursday, September 3 at 8:30 AM
Team assignment, 100 points max

Pascal scanner

The purpose of this assignment is to give you practice writing a scanner for Pascal.

Start with the `Scanner` and `Token` classes in [SimpleJava.zip](#) that we went over in class. Modify the classes to handle the following Pascal reserved word tokens:

```
PROGRAM BEGIN END REPEAT UNTIL WRITE WRITELN DIV MOD  
AND OR NOT CONST TYPE VAR PROCEDURE FUNCTION  
WHILE DO FOR TO DOWNTO IF THEN ELSE CASE OF
```

Handle the following Pascal special symbol tokens:

```
. , : := ; + - * / ( )  
= <> < <= > >= .. ' [ ] ^
```

Also recognize these tokens:

```
IDENTIFIER INTEGER REAL CHARACTER STRING END_OF_FILE ERROR
```

You can make any modifications that you deem necessary to the other classes. For a more complete list of Pascal tokens, see the syntax diagrams at <http://primepuzzle.com/tp2/syntax-diagrams.html>

Comments

Your scanner should treat each comment as it would treat a blank – comments should be ignored. Pascal comments are enclosed in curly braces { and }.

Strings and character literals

A literal Pascal string is enclosed in single quotes. If a single quote character is part of a string, it is represented by two consecutive single quotes. For example, 'It's' contains the characters It's. It is possible to have the empty string: ''

A literal Pascal character is simply a string with only one character. For example: 'a'

Test files

Test your code on test input file [Newton.txt](#):

```
PROGRAM Newton;

BEGIN
  writeln(' n   Square root');
  writeln('-----');

  FOR n := 1 TO 20 DO BEGIN
    write(n:2);

    root := n;
    prev := root;
    diff := 99999;

    WHILE diff > 0.000001 DO BEGIN
      root := (n/root + root)/2;
      diff := prev - root;
      prev := root;
    END;

    writeln(root:14:6)
  END
END.
```

Test input file [ScannerTest.txt](#) will give your scanner and token classes a good workout:

```
{This is a comment.}

{This is a comment
 that spans several
 source lines.}

Two{comments in}{a row} here

{Word tokens}
Hello world
begin BEGIN Begin BeGiN begins

{String tokens}
'Hello, world.'
'It's Friday!'
''
'A' 'x' ' ' ' '
' ' ' ' ' ' ' ' '
'This string
spans
source lines.'

{Special symbol tokens}
+ - * / := . , ; : = <> < <= >= > ( ) [ ] { } } ^ ..
+ -= <> <=> .....
```

Expected output

Your output for input file `ScannerTest.txt` should be similar to the following:

```
Tokens :

IDENTIFIER : Two
IDENTIFIER : here
IDENTIFIER : Hello
IDENTIFIER : world
        BEGIN : begin
        BEGIN : BEGIN
        BEGIN : Begin
        BEGIN : BeGiN
IDENTIFIER : begins
```

```

        STRING : 'Hello, world.'
        STRING : 'It's Friday!'
        STRING : ''
        CHARACTER : 'A'
        CHARACTER : 'x'
        CHARACTER : '''
        STRING : ' ' ' '
        STRING : ''''
        STRING : 'This string
spans
source lines.'
        PLUS : +
        MINUS : -
        STAR : *
        SLASH : /
        COLON_EQUALS : :=
        PERIOD : .
        COMMA : ,
        SEMICOLON : ;
        COLON : :
        EQUALS : =
        NOT_EQUALS : <>
        LESS_THAN : <
        LESS_EQUALS : <=
        GREATER_EQUALS : >=
        GREATER_THAN : >
        LPAREN : (
        RPAREN : )
        LBRACKET : [
        RBRACKET : ]
TOKEN ERROR at line 24: Invalid token at '}'
        ERROR : }
        CARAT : ^
        DOT_DOT : ..
        PLUS : +
        MINUS : -
        COLON_EQUALS : :=
        NOT_EQUALS : <>
        EQUALS : =
        LESS_EQUALS : <=
        EQUALS : =
        DOT_DOT : ..
        DOT_DOT : ..
        PERIOD : .
        INTEGER : 0
        INTEGER : 1
        INTEGER : 20
        INTEGER : 00000000000000000032
        INTEGER : 31415926
        REAL : 3.1415926
        REAL : 3.1415926535897932384626433
        PERIOD : .
        INTEGER : 14
TOKEN ERROR at line 32: Invalid number at '3.14.15926'
        ERROR : 3.14.15926
        IDENTIFIER : What
TOKEN ERROR at line 33: Invalid token at '?'
        ERROR : ?
TOKEN ERROR at line 34: String not closed at ''String
'not' closed'
        STRING : 'String 'not' closed

```

What to submit to Canvas

- A new version of `SimpleJava.zip` that includes your modified `Scanner` and `Token` classes.
- Text files of output from running your scanner on input files `Newton.txt` and `ScannerTest.txt`.

Submit to **Assignment #2: Pascal Scanner**

There should be only one submission per team.

Rubric

Your submission will be graded according to these criteria:

Criteria	Maximum points
Reserved words handled properly.	30
Special symbols handled properly.	30
Token errors handled properly.	30
Good output format.	10