San José State University
Department of Computer Science

CS/SE 153
Concepts of Compiler Design

Fall 2024
Instructor: Ron Mak

Assignment #2

Assigned: Thursday, August 29
Due: Thursday, September 5 at 4:00 PM
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 Simple.zip that we went over in class. It can already handle the following Pascal reserved word tokens:

`AND`	**`ARRAY`**	**`BEGIN`**	**`CASE`**	**`CONST`**
`DIV`	**`DO`**	**`DOWNTO`**	**`ELSE`**	**`END`**
`FILE`	**`FOR`**	**`FUNCTION`**	**`GOTO`**	**`IF`**
`IN`	**`LABEL`**	**`MOD`**	**`NIL`**	**`NOT`**
`OF`	**`OR`**	**`PACKED`**	**`PROCEDURE`**	**`PROGRAM`**
`RECORD`	**`REPEAT`**	**`SET`**	**`THEN`**	**`TO`**
`TYPE`	**`UNTIL`**	**`VAR`**	**`WHILE`**	**`WITH`**

It can also recognize these tokens:

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

Modify the classes **`Scanner`** and **`Token`** to recognize all the Pascal one-character and two-character special symbol tokens:

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

Class **Token** already has defined enumeration constants for them:

```
// Special symbols
PERIOD, COMMA, COLON, COLON_EQUALS, SEMICOLON,
PLUS, MINUS, STAR, SLASH, LPAREN, RPAREN,
EQUALS, NOT_EQUALS, LESS_THAN, LESS_EQUALS,
GREATER_THAN, GREATER_EQUALS, DOT_DOT, QUOTE,
LBRACKET, RBRACKET, CARAT
```

You can make any modifications that you deem necessary to the other classes.

Comments

Your **Scanner** class should treat each comment as it would treat a blank. Like blanks, comments should be skipped and ignored. Pascal comments are enclosed in curly braces **{** and **}**.

Strings and character literals

You will need to modify the s**tring()** method in class **Token.**

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'**.

Your scanner should distinguish between Pascal literal strings and characters.

Test files

First test your code on test input file **EmployeeListing.pas** from Assignment #1.

Then test input file **ScannerTest.txt** will give your **Scanner** and **Token** classes a good workout. It consists of Pascal tokens, often run together the way they would never appear in a valid Pascal program. Your scanner should still be able to extract them.

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

{Number tokens}
0 1 20 00000000000000000032  31415926
3.1415926  3.14159265358979323846264433 .14

{Bad tokens}
3.14.15926
What?
'String ''not'' closed
```

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 : 3.14159265358979323846264338
        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 **`Simple.zip`** that includes all your Java sources, including your modified **`Scanner`** and **`Token`** classes.
- Text files of output from running your scanner on the two input files **`EmployeeListing.pas`** 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.	20
Special symbols handled properly.	40
Token errors handled properly.	30
Good output format.	10