COP5556 Assignment 1

Implement a scanner for the programming language with the following lexical structure:

RawInputCharacter ::= any ASCII character

LineTerminator ::= LF | CR | CR LF

*LF is the ASCII character also known as “newline”, in java \n*

*CR is the ASII character also known as “return”, in Java, the char \r*

*CR immediately followed by LF counts as one line terminator, not two*

InputCharacter ::= RawInputCharacter, but not CR or LF

Input ::= (WhiteSpace | Comment | Token)\*

Token ::= Identifier | ReservedWord | Literal | ~~Separator | Operator~~ | Constant | Symbol

WhiteSpace ::= SP | HT | FF | LineTerminator

*SP is the ASCII character also known as “space”*

*HT is the ASCII character also known as “horizontal tab”*

*FF is the ASCII character known also known as “form feed”*

Comment ::= / / InputCharacter\* (LineTerminator | EOF) UPDATED 9/14

*EOF means end of input.*

Identifier ::= IdentifierChars but not a ReservedWord

IdentifierChars ::= IdentiferStart IdentifierPart\*

IdentifierStart ::= A..Z | a..z | \_ | $

IdentifierPart ::= IdentifierStart | Digit

Literal ::= IntLit | StringLit

IntLit ::= 0 | NonZeroDigit Digit\*

NonZeroDigit ::= 1 .. 9

Digit ::= NonZeroDigit | 0

StringLit ::= “ StringCharacter\* “

StringCharacter::= InputCharacter but not “ or \

| EscapeSequence

EscapeSequence ::=

\ b | \ t | \ n | \ f | \r | \” | \’ | \ \

Symbol ::= ( | ) | [ | ] | ; | , | ≪ | ≫ | = | > | < | ! | ? | : | == | != | <= | >=

| + | - | \* | / | % | -> | <- | @ | # | & | | UPDATED Add & and |

ReservedWord ::= X | Y | width | height | screen | screen\_width | screen\_height | image | int | string | red | green | blue

Constants ∷= Z | WHITE | SILVER | GRAY | BLACK | RED | MAROON | YELLOW | OLIVE | LIME | GREEN | AQUA | TEAL | BLUE | NAVY | FUCHSIA | PURPLE

* If an illegal character is encountered, your scanner should throw a LexicalException. The message should contain useful information about the error. The contents of the message will not be graded, but you will appreciate it later if it is helpful.
* If an integer literal is provided that is out of the range of a Java int, then your scanner should also throw a LexicalException. The contents of the error message will not be graded, but you will appreciate it later if it is helpful.
* Use the provided Scanner.java and ScannerTest.java as starting points. The provided code should compile but only the testEmpty test will succeed. All should succeed (plus many more!) with your completed Scanner.
* For string literals, the pos and length refer to characters in the char array that is providing the input to the scanner, including the delimiting “s and slashes for escape characters. The getText method removes the delimiters and handles the escapes.
* An EOF (end of file) token should be inserted as the last token. The position is as if it were an extra character at the end of the input; its length is zero.
* The language is case sensitive.