COMP 442 – Assignment 1

Lexical Analyzer

Gordon Bailey

9541098

I certify that this submission is my original work and meets the Faculty's Expectations of Originality

Lexical Specification

The following is the lexical specification which was used. The codes used in the implementation are the same as those shown in the table below.

Token	Lexeme
tok_id	distance, rate, time_of_day, x_1
tok_int_literal	0, 1, 2, 567, 100000
tok_float_literal	0.0, 0.1, 1.0, 3.14159, 2.56
tok_equals	==
tok_assignment	=
tok_diamond	<>
tok_less_than	<
tok_greater_than	>
tok_less_than_equals	<=
tok_greater_than_equals	>=
tok_plus	+
tok_minus	-
tok_star	*
tok_slash	/
tok_and	and
tok_or	or
tok_not	not
tok_semicolon	;
tok_comma	,
tok_dot	.
tok_open_paren	(
tok_open_brace	{
tok_open_square	[
tok_closeparen	)
tok_close_brace	}
tok_close_square	]
tok_if	if
tok_then	then
tok_else	else
tok_for	for
tok_class	class
tok_int	int
tok_float	float
tok_get	get
tok_put	put
tok_return	return

Potential Errors

The only error which is given is when an invalid character is encountered. All other situations can be resolved by breaking the encountered tokens into smaller subtokens which are lexically valid (although possibly not syntactically valid). The benefits of this approach is that no assumptions are made about what sequences of tokens are syntactically valid, therefore it is unlikely that the tokenizer will have to be modified very much to accommodate the

The error recovery method used is a “Panic Mode” technique, in which each invalid character is reported and skipped, until a valid character is encountered and the lexical analysis continues.

Implementation

The Scanner class was made using a “hand-written” implementation. This choice was made due to the greater ease in comprehending and debugging a hand written implementation versus a table-based one. The Scanner class keeps an instance of a class which extends the class State. All of these State subclasses implement methods to process an input stream until a new token is produced. Each token simply has 3 properties: an enum representing the token type, a string containing the value/lexeme of the token, and an integer containing the line number on which the token was encountered.

Input and Output

The input and output of the driver program is specified in the preferences file “settings.txt”. The input is taken from the file specified as the property “input”, and the output token stream is written to the file specified as “output”. Errors are reported to the programs standard error output.

Ambiguities

The following ambiguities were found in the given lexical definition.

Situation (example)	Potential Resolutions	Chosen resolution
123.123	tok_int_literal tok_dot tok_int_literal	tok_float_literal
	tok_float_literal	
0123	tok_int_literal tok_int_literal	tok_int_literal tok_int_literal
	Error (invalid token)	
123.1230	tok_float_literal tok_int_literal	tok_float_literal tok_int_literal
	Error (invalid token)	
123abc	tok_int_literal tok_id	tok_int_literal tok_id
	Error (invalid token)	
intabc	tok_int tok_id	tok_id
	tok_id	

A final ambiguity was whether or not to include tokens representing comments in the output. It was decided not to include them.