Lexical & Syntax Analysis

Conor Smyth, 12452382. All work is my own.

My implementation starts with the user code which initialises the parser and reads through the supplied file. You must supply the filename that you wish to analyse.

The next part of the analyser defines all the tokens for the language. It is broken into tokens to skip, keywords, identifiers, operators, braces and others. These recognise the valid tokens that the language accepts and is used by the syntax analysis. The comments are handled by accepting a single line comment with anything after excluding new lines, followed by a new line character, using regular expressions. The multi line comment is implemented in a similar fashion but instead excludes characters that end a comment block and then ends with a end comment block character, also implemented using regular expressions. The keywords are the keywords defined by the language which are special words that need to be recognisable. An id is checked by ensuring it begins with a letter followed by any combination of characters, numbers and underscores. The punctuation and braces are trivial and simply match with the values which are significant for the language.

Next is the grammar. I used the grammar described on the assignment page. I had issues with left-recursion so I modified the rules to remove the left recursion issues. The expression rule is broken down into expression, term and fragment to avoid left-recursion. The condition rule takes into account parenthesis and multiple conditions together and uses another rule to help remove the left-recursion error.

The analyser was tested against the sample file supplied and two other test files. The javacc file compiles with one warning, no errors and analyses the files successfully