# Code Hierarchy

## Classes

* Compiler – The main class for the project.
* Lexer - represents the lexical analyzer, this is the output of for the Jflex.jar.
* LexicalError - represents the lexical exceptions thrown during analysis process with the Lexer.
* Sym – contains a list of legal tokens in the language.
* Token – extends Symbol, represents a token in the analysis.
* TokenWithIdentifier – represents an identifier token.

## Known bugs

* None

# Code Structure and testing strategy

## Regular expressions

### Macros:

**Comments**

- Comment = {TraditionalComment} | {EndOfLineComment} | {DocumentationComment}

TraditionalComment = "/\*" [^\*] ~"\*/" | "/\*" "\*"+ "/"

EndOfLineComment = "//" {InputCharacter}\* {LineTerminator}?

DocumentationComment = "/\*\*" {CommentContent} "\*"+ "/"

CommentContent = ( [^\*] | \\*+ [^/\*] )\*

**Special chars**

LineTerminator = \r|\n|\r\n

InputCharacter = [^\r\n]

WhiteSpace = {LineTerminator} | [ \t\f]

**Simple Regex**

ALPHA=[A-Za-z]

DIGIT=[0-9]

ALPHA\_NUMERIC={ALPHA}|{DIGIT}

DecIntegerLiteral = 0 | [1-9][0-9]\*

PrintableAsciiChar = [ -~] - all ascii chars between 32 - 126

**Identifiers**

VariableIdentifier = [a-z]({ALPHA\_NUMERIC})\*

ClassIdentifier = [A-Z]({ALPHA\_NUMERIC})\*

### States:

* **Initial** – initial state
* **String** – state for accepting strings, first " will change the state to string, second " will change the state back to initial state.

### Regular expression for all the tokens out code recognizes:

|  |  |
| --- | --- |
| "=" | sym.ASSIGN |
| "boolean" | sym.BOOLEAN |
| "break" | sym.BREAK |
| "class" | sym.CLASS |
| "," | sym.COMMA |
| "continue" | sym.CONTINUE |
| "/" | sym.DIVIDE |
| "." | sym.DOT |
| "==" | sym.EQUAL |
| "extends" | sym.EXTENDS |
| "else" | sym.ELSE |
| ">" | sym.GT |
| ">=" | sym.GTE |
| "if" | sym.IF |
| "int" | sym.INT |
| "&&" | sym.LAND |
| "[" | sym.LB |
| "(" | sym.LP |
| "{" | sym.LCBR |
| "length" | sym.LENGTH |
| "new" | sym.NEW |
| "!" | sym.LNEG |
| "||" | sym.LOR |
| "<" | sym.LT |
| "<=" | sym.LTE |
| "-" | sym.MINUS |
| "%" | sym.MOD |
| "\*" | sym.MULTIPLY |
| "!=" | sym.NEQUAL |
| "null" | sym.NULL |
| "+" | sym.PLUS |
| "]" | sym.RB |
| "}" | sym.RCBR |
| ")" | sym.RP |
| ";" | sym.SEMI |
| "return" | sym.RETURN |
| "static" | sym.STATIC |
| "this" | sym.THIS |
| "true" | sym.TRUE |
| "false" | sym.FALSE |
| "void" | sym.VOID |
| "while" | sym.WHILE |
| "string" | sym.string |
| {DecIntegerLiteral} | sym.INTEGER |
| {VariableIdentifier} | sym.ID |
| {ClassIdentifier} | sym.CLASS\_ID |

## Testing Strategy

* Tests on strings
* Test on and compare with example output files
* Test comments
* Test on invalid files