Design File

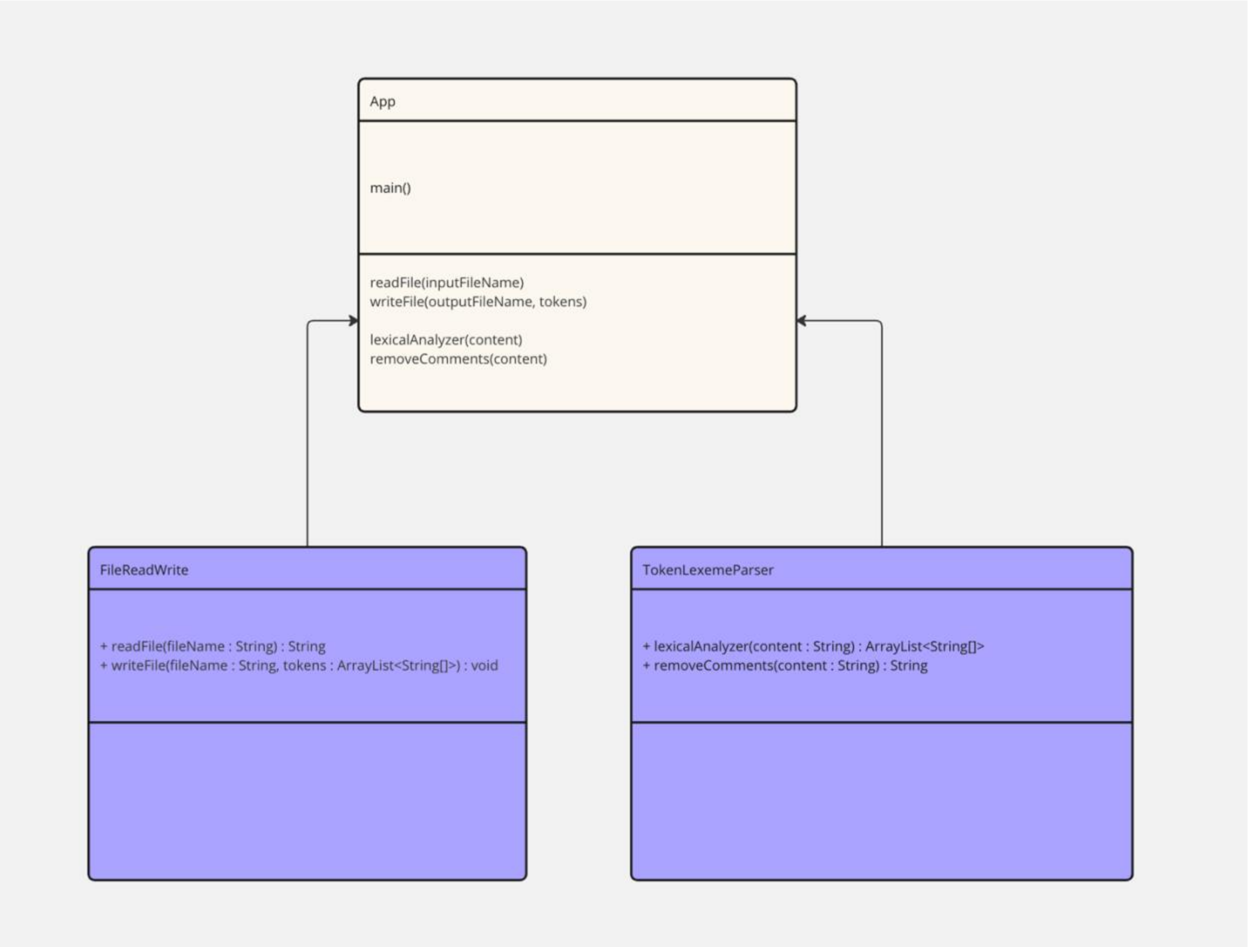
CPSC 323 Project 1

Contributors:

1. Guido Asbun
2. Cade Duncan
3. Briyana Verdugo

Our program was designed and implemented using Java Oracle version 21.

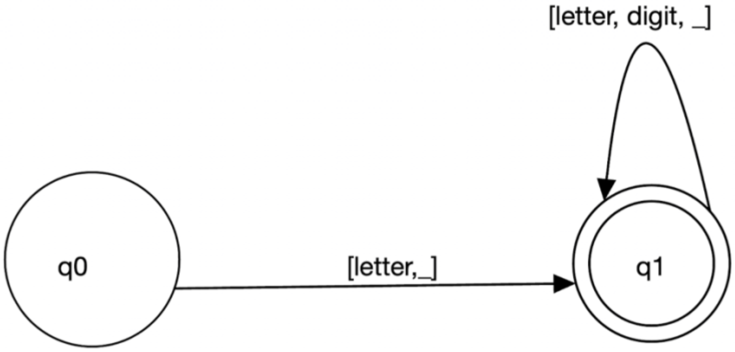
Class Diagrams:


App
main()
readFile(inputFileName)
writeFile(outputFileName, tokens)
lexicalAnalyzer(content)
removeComments(content)
FileReadWrite
+ readFile(fileName : String) : String
+ writeFile(fileName : String, tokens : ArrayList<String[]>) : void
TokenLexemeParser
+ lexicalAnalyzer(content : String) : ArrayList<String[]>
+ removeComments(content : String) : String

Logic:

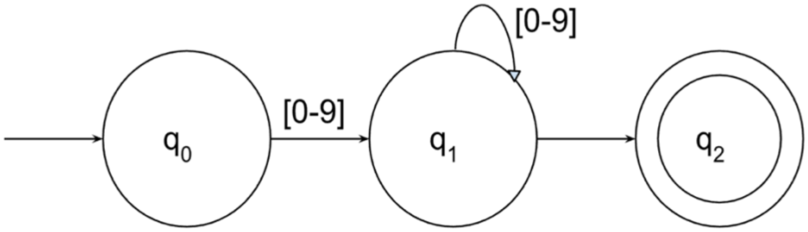
- All the primary logic to parse and tokenize the lexemes resides in the lexicalAnalyzer() function in the TokenLexemeParser class.
    1. lexicalAnalyzer() receives the string-afied content passed from main()
    2. Creates a Java Regular Expressions in string format that can be broken down into 5 different types of tokens:
        a. Keywords
        b. Separators
        c. Identifiers
        d. Real and Integer Numbers
        e. Strings
    3. The String of tokens is then converted into a pattern using the Java pattern.compile built in function
    4. The pattern is then separated into searchable groups with the Java .match function
    5. Lastly, the function searches through the groups and adds to the array in the form of [“token”, “lexeme”].

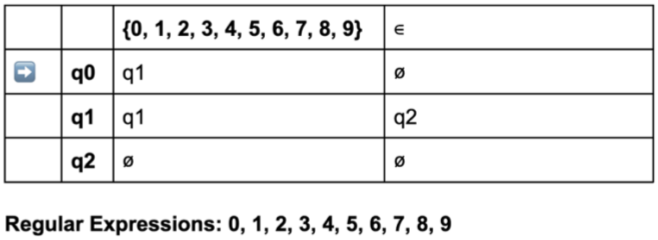
FSA for Tokens:
Identifier:



	Input transition	Next State
q0	letter or underscore	q1 (identifier)
q1	letter, digit, or underscore	q1 (identifier)

Integer:





		{0, 1, 2, 3, 4, 5, 6, 7, 8, 9}	∊
➡️	**q0**	q1	ø
	q1	q1	q2
	q2	ø	ø

Regular Expressions: 0, 1, 2, 3, 4, 5, 6, 7, 8, 9

Sample input_scode.txt:

```
while (k < lower) s = 33.00;
while (k < upper) {
    if (s < 33.00) {
        s = 33.00;
        // This should be ignored
        }
}

class A {
    k = [k + 1];
    String s = "Hello";
    s = s + 1.00; // This should be ignored
    System.out.println(s);
}
// This should be ignored
```

Sample output_scode.txt:

```
Token        Lexeme
--------------------------------
Keyword       while
Separator      (
Identifier    k
Separator      <
Identifier    lower
Separator      )
Identifier    s
Separator      =
Real         33.00
Separator      ;
Keyword       while
Separator      (
Identifier    k
Separator      <
Identifier    upper
Separator      )
Separator      {
Keyword        if
Separator      (
Identifier    s
Separator      <
Real         33.00
Separator      )
Separator      {
Identifier    s
Separator      =
Real         33.00
Separator      ;
Separator      }
Separator      }
Keyword       class
Identifier    A
Separator      {
Identifier    k
Separator      =
Separator      [
Identifier    k
Integer      1
Separator      ]
Separator    ;
Keyword      String
Identifier   s
Separator    =
String       "Hello"
Separator    ;
Identifier   s
Separator    =
Identifier   s
Real         1.00
Separator    ;
Keyword      System
Separator    .
Keyword      out
Separator    .
Keyword      println
Separator    (
Identifier   s
Separator    )
Separator    ;
Separator    }
```