File - /Users/JH/Documents/GitHub/NTU_ComplierTech_Lab/lab1/Lab1/src/test/LexerTests.java

```
1 package test;
 2
 3 import static org.junit.Assert.*;
 5 import java.io.IOException;
 6 import java.io.StringReader;
 8 import lexer.Lexer;
 9
10 import org.junit.Test;
11
12 import frontend. Token;
13 import frontend. Token. Type;
14 import static frontend. Token. Type. *;
15
16 /**
17
    * This class contains unit tests for your lexer. Currently, there is
   only one test, but you
18
   * are strongly encouraged to write your own tests.
19
    */
20 public class LexerTests {
       // helper method to run tests; no need to change this
21
22
       private final void runtest(String input, Token... output) {
23
           Lexer lexer = new Lexer(new StringReader(input));
24
           int i=0;
25
           Token actual, expected;
           try {
26
27
               do {
28
                    assertTrue(i < output.length);</pre>
29
                    expected = output[i++];
30
                    try {
31
                        actual = lexer.nextToken();
32
                        assertEquals(expected, actual);
33
                    } catch(Error e) {
                        if(expected != null)
34
35
                            fail(e.getMessage());
36
                        return;
                    }
37
               } while(!actual.isEOF());
38
           } catch (IOException e) {
39
40
               e.printStackTrace();
41
               fail(e.getMessage());
42
           }
43
       }
44
       /** Example unit test. */
45
46
       @Test
47
       public void testKWs() {
48
           // first argument to runtest is the string to lex; the
   remaining arguments
```

```
49
            // are the expected tokens
50
            runtest("module false return while boolean break else if
   import int public true type void",
                    new Token(MODULE, 0, 0, "module"),
new Token(FALSE, 0, 7, "false"),
51
52
                    new Token(RETURN, 0, 13, "return"),
53
                    new Token(WHILE, 0, 20, "while"),
54
                    new Token(BOOLEAN, 0, 26, "boolean"),
55
                    new Token(BREAK, 0, 34, "break"),
56
                    new Token(ELSE, 0, 40, "else"),
57
                    new Token(IF, 0, 45, "if"),
58
59
                    new Token(IMPORT, 0, 48, "import"),
                    new Token(INT, 0, 55, "int"),
60
                    new Token(PUBLIC, 0, 59, "public"),
61
                    new Token(TRUE, 0, 66, "true"),
62
                    new Token(TYPE, 0, 71, "type"),
63
                    new Token(VOID, 0, 76, "void"),
64
                    new Token(EOF, 0, 80, ""));
65
66
       }
67
       public void testOperators() {
68
            runtest("/ == = >= > <= < - != + *",
69
                    new Token(DIV, 0, 0, "/"),
new Token(EQEQ, 0, 2, "=="),
70
71
                    new Token(EQL, 0, 5, "="),
72
                    new Token(GEQ, 0, 8, ">="),
73
                    new Token(GT, 0, 11, ">"),
74
                    new Token(LEQ, 0, 14, "<="),
75
                    new Token(LT, 0, 17, "<"),
76
                    new Token(MINUS, 0, 19, "-"),
77
78
                    new Token(NEQ, 0, 21, "!="),
                    new Token(PLUS, 0, 24, "+"),
79
                    new Token(TIMES, 0, 26, "*"),
80
                    new Token(EOF, 0, 28, ""));
81
       }
82
83
84
       @Test
       public void testIDs() {
85
            runtest("aSdf923k_dsf2145",
86
                    new Token(ID, 0, 0, "aSdf923k_dsf2145"),
87
                    new Token(EOF, 0, 16, ""));
88
            runtest("___aSdf923k_dsf2n45",
89
                                            ___aSdf923k_dsf2n45"),
                    new Token(ID, 0, 0, "_
90
                    new Token(EOF, 0, 19, ""));
91
            runtest("_",
92
                    new Token(ID, 0, 0, "_"),
93
                    new Token(EOF, 0, 1, ""));
94
       }
95
96
97
       @Test
```

File - /Users/JH/Documents/GitHub/NTU_ComplierTech_Lab/lab1/Lab1/src/test/LexerTests.java 98 public void testStringLiteralWithDoubleQuote() { 99 runtest("\"\"\"", 100 new Token(STRING_LITERAL, 0, 0, ""), (Token)null); 101 } 102 103 104 @Test public void testStringLiteral() { 105 106 runtest("\"\\n\"". new Token(STRING_LITERAL, 0, 0, "\\n"), 107 new Token(EOF, 0, 4, "")); 108 109 110 } 111 112 @Test 113 public void testStringLiteralWhitespace() { runtest("\" foo_ % ^&_trump says module\"", 114 115 new Token(STRING_LITERAL, 0, 0, " foo_ % ^&_trump says module"), new Token(EOF, 0, 32, "")); 116 117 } 118 119 120 @Test 121 public void testIntLiteral() { runtest("0000123000", 122 new Token(INT_LITERAL, 0, 0, "0000123000"), 123 new Token(EOF, 0, 10, "")); 124 125 126 } 127 128 @Test public void testPunctuation() { 129 runtest("if (foo[0] == bar[1]) { $method(1, \trump\tru$ 130 new Token(IF, 0, 0, "if"), 131 new Token(LPAREN, 0, 3, "("), 132 new Token(ID, 0, 4, "foo"), 133 new Token(LBRACKET, 0, 7, "["), 134 new Token(INT_LITERAL, 0, 8, "0"), 135 136 new Token(RBRACKET, 0, 9, "]"), 137 new Token(EQEQ, 0, 11, "=="), new Token(ID, 0, 14 , "bar"), 138 new Token(LBRACKET, 0, 17, "["), 139 new Token(INT_LITERAL, 0, 18, "1"), 140 new Token(RBRACKET, 0, 19, "]"), 141 new Token(RPAREN, 0, 20, 142 143 new Token(LCURLY, 0, 22, "{"), new Token(ID, 0, 23, "method"), 144 new Token(LPAREN, 0, 29, "("), 145 new Token(INT_LITERAL, 0, 30, "1"), 146

File - /Users/JH/Documents/GitHub/NTU_ComplierTech_Lab/lab1/Lab1/src/test/LexerTests.java

```
new Token(COMMA, 0, 31, ","),
new Token(STRING_LITERAL, 0, 33, "trump"),
147
148
                                new Token(RPAREN, 0, 40, ")"),
new Token(RCURLY, 0, 41, "}"),
new Token(SEMICOLON, 0, 42, ";"),
149
150
151
                                 new Token(EOF, 0, 43, ""));
152
153
             }
154
155
156
157 }
158
```